ELARA:
Environmental Liaison
and Automated Recycling Assistant

Thesis submitted in partial fulfillment of the requirement for

Honors in

Computer Science

Marcela S. Melara

Adviser, John Vaughn **April 10, 2012**

Acknowledgments

I would like to thank my adviser John Vaughn for his enthusiasm and his invaluable support from beginning to end. Working with him has been a huge learning experience, and his wit and out-of-the-box ideas helped me take this project to a much higher level that I had originally envisioned. I am glad to have been able to work with such a flexible and visionary adviser.

Furthermore, I would like to give thanks to Marc Corliss who introduced me into the world of computer science research. It is because of the opportunity he gave me to work on his research project as an undergraduate student that I will begin pursuing a Ph.D. in Computer Science in the Fall of 2012.

Last but not least, I would like to thank my family, Wade, and my friends for always being there when I needed advice or just a place to vent while working on this project. I owe the fact that I have kept part of my sanity to you.

Contents

List of Figures

1

Introduction

1.1 Motivation for this Project

Growing up in Germany, a country where sustainability is at the center of people's lives, shaped my view on how to lead an environmentally aware life. In particular, Germans pay a lot of attention to waste sorting. In most cities, municipal waste is separated by compostables, recyclables and landfill garbage, not to mention the degree to which recyclables are separated. For instance, there are recycling collection sites which separate glass alone into green, brown and clear glass.

The waste separation policies in Germany are very systematic, as can be seen in Fig. 1.1. Similar figures can be found at other schools or other institutions across Germany. It is used at the Gymnasium Schloss Neuhaus to help teachers and students identify which items belong in which bins. This is a significant fact of life to both urban and rural citizens.

When I moved back to the U.S. five years ago, I realized that recycling and waste sorting is not a common practice. After having been used to minimizing how much waste goes to the landfill in Germany, it was hard for me to not separate my garbage at home and accept the fact that many recyclable items will end up in landfills.

Figure 1.1: What belongs where? Image courtesy of www.gymnasium-schloss-neuhaus.de

To put this into perspective, in 2010 the EPA recorded that more than half (54.5%) of our total waste is recyclable (paper, plastic, glass and metals) (see Fig. 1.2); however, we only recycled 42.6% of all recyclable materials that year [4]. This means that we only recycled about a quarter of our total waste, half of what is possible.

One of the reasons I choose to come to Hobart and William Smith Colleges (HWS) was the on-campus efforts to help the environment through the Campus Greens student club and the HWS Goes Green initiative. In particular, I was attracted by the efforts being made to increase recycling and waste sorting. Waste bins for each type of garbage have been installed in and around all dining services buildings, and even in a few academic buildings. Unfortunately, I soon noticed that even with these separate waste bins on campus, students often do a poor job of identifying and sorting recyclable materials.

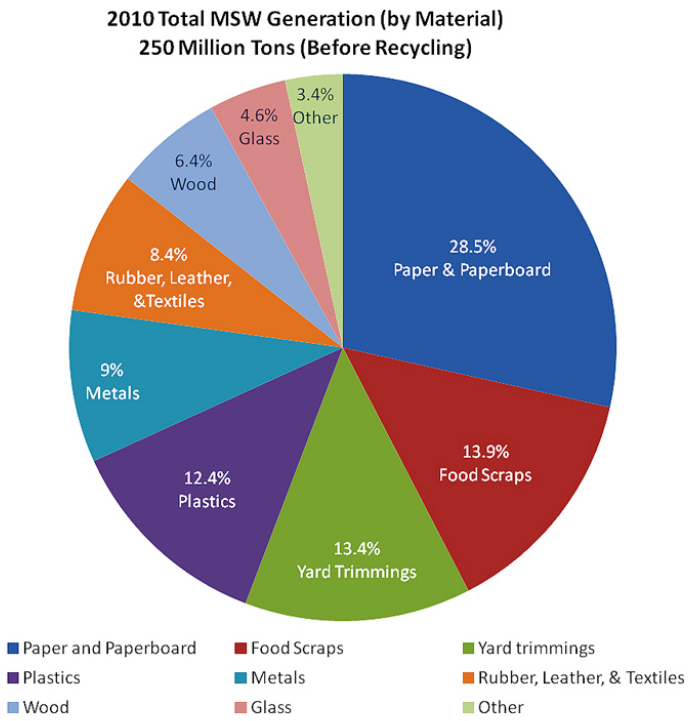

Figure 1.2: EPA Municipal Solid Waste (by Material). This figure shows the percent generation of municipal solid waste by material recorded in 2010. Image courtesy of www.epa.gov

While various signs have been put up adjacent to bins to illustrate the items that belong in each of the bins, I have identified two major flaws with these guidelines: (1) The signs are outdated since the items sold at the various dining facilities change with a certain frequency, and (2) The signs are not detailed enough since they omit many common items that students purchase at the dining facilities. Even if these two flaws could be corrected, this system is still very ineffective since there is no way of verifying that all the items in a particular bin have been discarded properly. More than once have I seen cleaning staff discard all three bins into the landfill dumpster because waste items have not been sorted properly.

For this reason, I began to look at hardware and software solutions to both automate this process and help reduce these garbage sorting errors. Building on my experience in the Embedded Computing course in 2009, in which I designed a very

simple machine that would sort inserted waste items into the proper bins, I designed and implemented an end-to-end system. My solution combines several features of reverse vending machines, ATMs and other service kiosks, while also taking advantage of crowdsourcing and social media. I have created ELARA, a hardware and software system designed to further improve the recycling rates on the HWS campus.

Chapter 1 of this thesis describes ELARA, background information about recycling is in Chapter 2, and the concept of crowdsourcing in Chapter 3. Chapter 4 contains a detailed description of the design and implementation of ELARA, while Chapter 5 discusses potential future work, and I offer reflections and conclusions in Chapter 6.

1.2 ELARA Overview

ELARA, the Environmental Liaison and Automated Recycling Assistant, is a new system that facilitates recycling and waste sorting by helping people identify the items which are recyclable and those which are not. The most immediately noticeable aspect of ELARA is a networked kiosk to help users sort their waste correctly; the kiosk can replace the existing waste stations in many facilities. The kiosk is the front end of a larger system of hardware and software.

The ELARA system is comprised of four major elements, which I developed concurrently:

1. A database containing information about waste items, users and kiosks
2. The networked hardware for the kiosk allowing user interaction with ELARA
3. The kiosk client software which communicates with the ELARA database
4. The ELARA website communicating with users online

These four components together create a network of stand-alone embedded devices, users and the database server. The idea is that the kiosk software, running on a touch panel computer, communicates with the database on the server. A user

interacts with the kiosk about recycling choices, and the system records each user transaction in the database. The user can then visit the website and view her recycling history, suggest improvements and possibly contribute to the project.

The kiosk has been built modeled after various other service kiosks. Fig. 1.3 shows a schematic of the proof-of-concept ELARA kiosk.

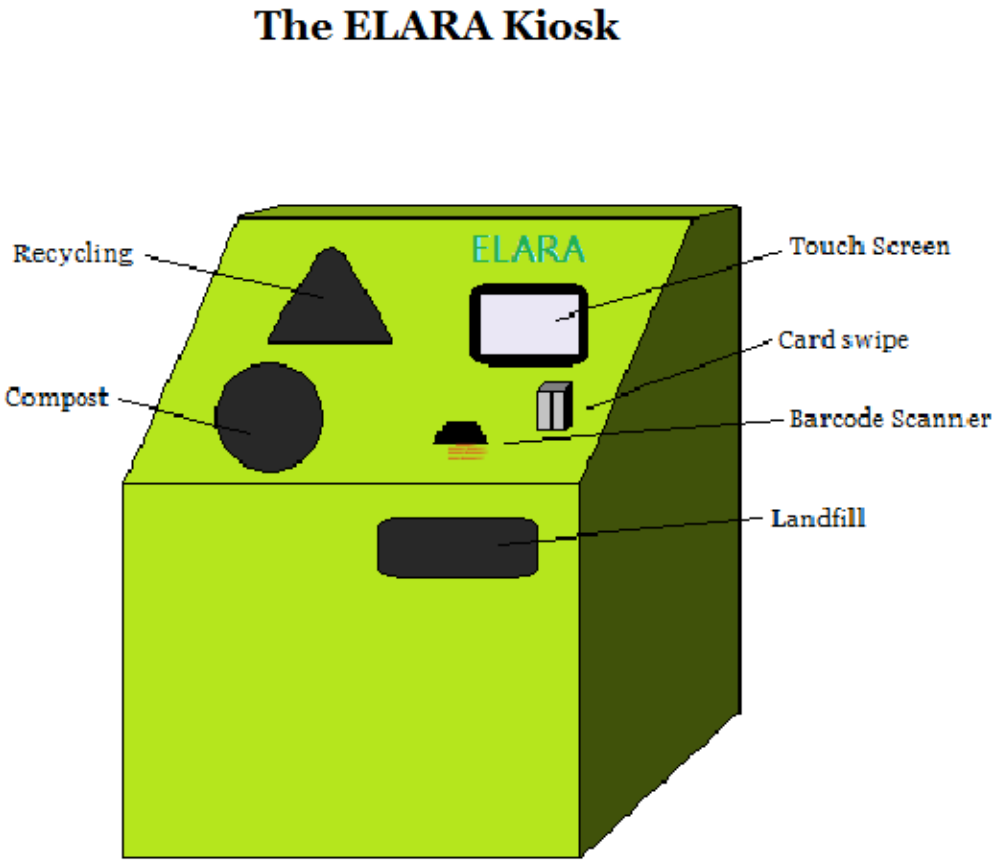

Figure 1.3: Schematic of the proof-of-concept ELARA kiosk

Here is how a user interacts with the kiosk:

1. The touch panel computer displays a welcome screen, asking the next user to select if she is a registered user or a guest user.
2. When the user is ready to use the kiosk, she presses the appropriate button on the touch screen. Registered users then swipe their user ID card using the card swipe to sign in.

3. Once the user has been verified, the screen will show a main menu. The user can press the "Start Recycling!" button on the screen to begin.
4. The user may scan her item's barcode using the barcode scanner at the kiosk.
5. The kiosk will verify the item and indicate the proper bin for the user's item.
6. After the user has inserted her item into the proper bin, she may continue with another item, or end her session.

ELARA improves upon existing recycling machines in two main ways:

1. By combining familiar features of reverse vending machines, ATMs and other common service kiosks to create a recognizable, user-friendly kiosk.
2. Through crowdsourcing, i.e., by allowing any person to contribute their knowledge about recyclables and waste items, and storing this information in the database.

This proof-of-concept system is designed to be extensible in several ways and allows customization by users, kiosk managers, and administrators. Some enhancements and future extensions are discussed in the following narrative. While not a primary goal of the project, ELARA can be a viable commercial design with additional investment in time and money.

2

Prior Work

In order to create a recognizable look and feel and make the kiosk more user-friendly and intuitive to use, I modelled the ELARA kiosk after three existing types of service kiosks. These can be divided into the following three categories:

1. Reverse Vending Machines
2. Bank ATMs
3. Other service kiosks (e.g. flight check-in kiosks, self-checkout kiosks etc.)

2.1 Reverse Vending Machines

A Reverse Vending Machine (RVM) is a machine which accepts empty, recyclable plastic bottles and/or aluminum cans, and returns the deposit to the user for each bottle or can dispensed. As the name suggests, Reverse Vending Machines, contrary to ordinary vending machines which dispense a full beverage or food container in exchange for a small payment, accept empty beverage receptacles and return a refund with the value of the total deposit. These machines come in many shapes; some machines only accept plastic bottles, others only alumnium cans, others both, some have small touch screen panels as their user interface, others just have buttons, some even take care of crushing the returnable to save space in the bin. What almost all

RVMs have in common is that they scan an item's barcode to identify it, and all print out a small voucher with the total amount of deposit money for the returned items.

The idea of reverse vending machines has existed since the 1970's when Petter and Tore Planke, the founders of the Norweigian company Tomra, began working on a solution to the following problem: A Norwegian supermarket was having trouble handling all the returned empty bottles, since there were too many and their procedure was not very organized [11].

After Tomra's first version of an RVM, other companies such as Wincor Nixdorf from Germany and Envipco from the U.S. followed them into the market [19]. The most recent popular addition to the RVM market has been Australia's Envirobank. This RVM has a touch screen panel and two buttons as its user interface. Users simply insert any empty bottle or can into the round slot, and its barcode is scanned while in the slot. They press a button on the machine to display their refund possibilities and they select their choice by touching the proper image on the screen [8].

Issues that arise with ordinary RVMs is that they require all receptacles and returnables to have barcodes. While this is not an issue I am fully addressing with ELARA, there are problems as not all waste items have barcodes that will make them easily identifiable. Nevertheless, since all beverage returnables have barcodes, this is not an issue for RVMs. However, this is a limitation in RVMs; these machines are very limited in the kinds of waste items they accept, so a single ELARA kiosk expands on that by accepting recyclables, compostable waste and landfill garbage. As a beginning design, I will also be focusing on beverage containers but plan on expanding the range of items accepted by the kiosk. In the future, ELARA's acceptance range can be expanded indefinitely even for items without barcodes.

An even larger issue with reverse vending machines is that they do not have a connection with the recycling profile of their users. If RVMs collected recycling data for their users, beverage companies could use this information to encourage their consumers to buy more beverages, and researchers could use the collected data

to carry out extensive studies of recycling trends and research ways to incentivize recycling. As mentioned above, ELARA improves upon ordinary RVMs by including a data collection mechanism in the kiosk touch panel software application and in the database.

2.2 ATMs

An automatic or automated teller machine, better known as an ATM, is a machine which allows its users to perform bank transactions from a remote location. These machines usually have a small screen with buttons allowing the selection of a menu of choices on the screen and also a numeric key pad to enter PIN numbers and the amount of money to withdraw or deposit. Each user proffers a debit or credit card which, in combination with the PIN number, is how users identify themselves at the ATM and gain access to their account. I use the ATM as a model for user authentication in ELARA since each registered user will need an HWS ID card to identify themselves by swiping this card at the kiosk. This will suffice for this pilot project, users will not need to also enter a unique PIN number.

From my personal experience, ATMs have a common set of user interactions. When the ATM is idle, there is a generic welcome screen which asks the user to insert their card and then enter their PIN number to begin a session with the machine. Once the user has been authenticated, the screen then displays a number of main actions the user can take with the machine; my local ATM lets the user choose between checking their account balance, withdrawing money from a checking account, withdrawing money from a savings account, and depositing money to an account. At the end of one such action, for instance withdrawal of money from a checking account, the ATM will prompt the user to continue or to end her session. A continuation of the session takes the user back to the main menu while, ending the session displays a "Thank You" screen before going back to the welcome screen.

I decided to imitate the ATM for its stepwise user interactions. On the one hand, since it helped me devise a pratical and efficient user interaction pattern, and on the other hand, it helps the interactive design because many people are already familiar with the format of ATMs. Therefore, I developed a stepwise set of user interactions very similar to those of an ATM. I shall explain ELARA's user interactions in Chapter 4.

One issue that designers of ATMs face in dealing with such high-value information is security. The software running on the ATM must allow secure network connections that cannot be attacked by malicious users who want to steal the bank account information of others. While ELARA at this point does not send high-value information across a network, there is still some personal information, namely recycling profiles, being sent that should not be stolen. In particular, for some features that ELARA may have in the future, network security such as that of ATMs will be necessary. Chapter 5 will elaborate on future work.

2.3 Other service kiosks

With the increasing ubiquity of technology in many aspects of our lives, service kiosks have become very common in many locations. Examples include self-service flight check-in kiosks in airports, information kiosks in businesses, and self-checkout kiosks in grocery stores. All of these types of kiosks have a graphical user interface on a screen, most usually a touch panel. These allow easy navigation through the screens and processes, giving the kiosk a friendly feel. The user interactions for service kiosks are also organized into a stepwise process.

ELARA follows these service kiosks as examples for both its physical and its user interfacce design. In particular, I was looking for a user-friendly design, that would be attractive to many users, unlike ATMs which have a very bland user interface. Some kiosks include images and buttons on their user interface (UI) making

the kiosk a truly visual experience as well. Additionally, many service kiosks need other external hardware, for instance a barcode scanner, which ELARA will also feature. This last group of service kiosks besides RVMs provided additional resources for design and implementation ideas which guided the creation of ELARA.

More social aspects of ELARA's design come from the recent proliferation of social media on the web. This aspect of the system design is discussed in the next chapter.

3

Crowdsourcing

Crowdsourcing is the concept that larger groups of people can provide expertise and insights that have traditionally been associated with a specialist or a small group of experts. Crowdsourcing has emerged as a viable solution for businesses, researchers, the military, and others to gather knowledge and expertise that previously might have gone unnoticed [5]. Crowdsourcing systems are used to help solve a wide variety of problems, so it only makes sense that numerous such systems have appeared on the Internet over the past decade, where millions of humans can come together to contribute to a project [1].

The idea of crowdsourcing first arose in 2001 with the start of Threadless, a company with a new kind of business plan: Anybody can submit designs for graphic T-shirts, other users vote for the best design and the winning T-shirt is put on sale [6]. The winning designers say that participation in these design contests is not only about the money but about the "the emerging reputation economy, where people work late night shifts to work on a creative endeavor or another in the hope that their community... acknowledge their contribution" [6].

Leitmeister *et al.* called this phenomenon an "Ideas Competition", which is characteristic of most crowdsourcing systems [7]. The authors were able to discern common trends in ideas competitions. "Typically, the tasks given to the participants

are kept generic, offering them a broad platform on which they can base their ideas. Submissions in the initial phases of ideas competitions include a brief description of ideas usually limited to five pages. Incentives for participation often comprise cash prices. The evaluation process is carried out by juries and the typical duration is between 4 and 26 weeks" [7]. One main characteristic of ideas competitions, especially in our technology-centric culture, is that most of them take place on the Internet; the competition is publicized online, and ideas can also be submitted online [7].

In a more general context, it seems that many crowdsourcing systems have a competition component to them. Competition is going on constantly. Even on social networking sites where users can socialize with their friends, people are also competing for bragging rights in various domains. For example, from my personal experience on Facebook, I have seen competitions occur in many different ways. One friend of mine recently messaged all her friends to "like" a certain business' Facebook page and tell them that we had been referred to that page by her. If she could reach a certain number of references, the business would offer to hire her. The bragging rights were for my friend to show that she had so many friends who all wanted her to get that job.

These characteristics of crowdsourcing gave me a sense of how my design for ELARA had to be to attarct a wide range of users, and to give them incentives to contribute knowledge to ELARA as well as to continue using ELARA on a regular basis.

3.1 Crowdsourcing in ELARA

ELARA's design also has a social aspect, which comes from the recent proliferation of social communities on the web. One very important driver of these successful businesses has been widespread participation of users in providing information, suggestions for improvements, and social interactions among larger groups.

Typical use of crowdsourcing in ELARA comes from the collection of knowledge about recyclables and other waste items. The ELARA database has been designed to keep record of waste items, and the idea is that any person can go onto the ELARA website, view the contents of the database to see which items are already on record, and if there is an item they do not see in the database yet, but they believe belongs on record, they can fill out a form on the website to enter this item into the database.

Another option that users have to contribute their knowledge about recyclables and other waste items is to vote for or against the recyclability of an existing item in the database. This can also be done via the ELARA website. Having users cast votes on items will increase the accuracy of the information in the database since it will reflect the knowledge of a larger group of people instead of that of a smaller group of experts. For example, say that a plastic water bottle is on record in the database. If one person votes on this item being recyclable, the chance of this item truly being recyclable is rather low, but if 100 people vote on this item being recyclable, the probability that this is accurate is a lot higher. Thus, the information in the database becomes more accurate. Since the kiosk uses this information in the database to assist its users in disposing of their waste correctly, higher accuracy in the information improves the assistance that the kiosk can give its users.

Nevertheless, by having any person add items into the database and cast votes, a lot of information in the database is bound to be inaccurate. Therefore, ELARA also has a built-in mechanism for users to perform quality checks on the contributions of other users. Since the kiosks record whether or not a user has just recycled or not, registered users can accumulate recycling points which partially shows that they recycle a lot and thus probably be more knowledgeable about recycling. If each new item contribution to the database also indicates who added the item, these users can be up-voted or down-voted depending on the accuracy of their contribution. What this means is that, if a user has contributed accurate information and other

knowledgeable users up-vote her, she is more likely to be trusted when she makes other contributions. At the same time, if a user makes an inaccurate contribution, and other knowledgeable users see this bad information, they can down-vote this user to indicate that she is not trustworthy. Systems of trust such as this one are very popular in crowdsourcing sites such as online assistance forums, and can help other users attain the most accurate information possible.

Crowdsourcing, however, is not only about information and knowledge collection. Social networks are also crowdsourcing systems. I shall elaborate more on this aspect of ELARA in Section 4.6. The technical aspects of ELARA are described in the next chapter.

4

ELARA Design and Implementation

The ELARA System is comprised of four major elements, which shall be described in detail in this chapter. When I began to work on ELARA, the first properties I thought of were that the kiosk should be attractive and easy to use. Given my interest in computer networks, I also added the property that my kiosk be networked; I envisioned that multiple such kiosks could be placed at various remote locations and all still draw on the same information following the client-server model. This would create a distributed system made up of the kiosks (clients) and a central server; the client-server model is particularly appropriate since the kiosks will be making requests for information, which the server receives, processes and sends the requested information back to the client [13]. This server enables the construction of a distributed system of multiple kiosks networked to the server, and multiple users at home visiting the ELARA website. In the future, mobile users might also become part of the distributed system by connecting to the server through an ELARA smartphone application. Fig. 4.1 shows an overview of the ELARA system.

Note: In order to reduce space in this thesis, the ELARA source code is available in the accompanying media, and from the author.

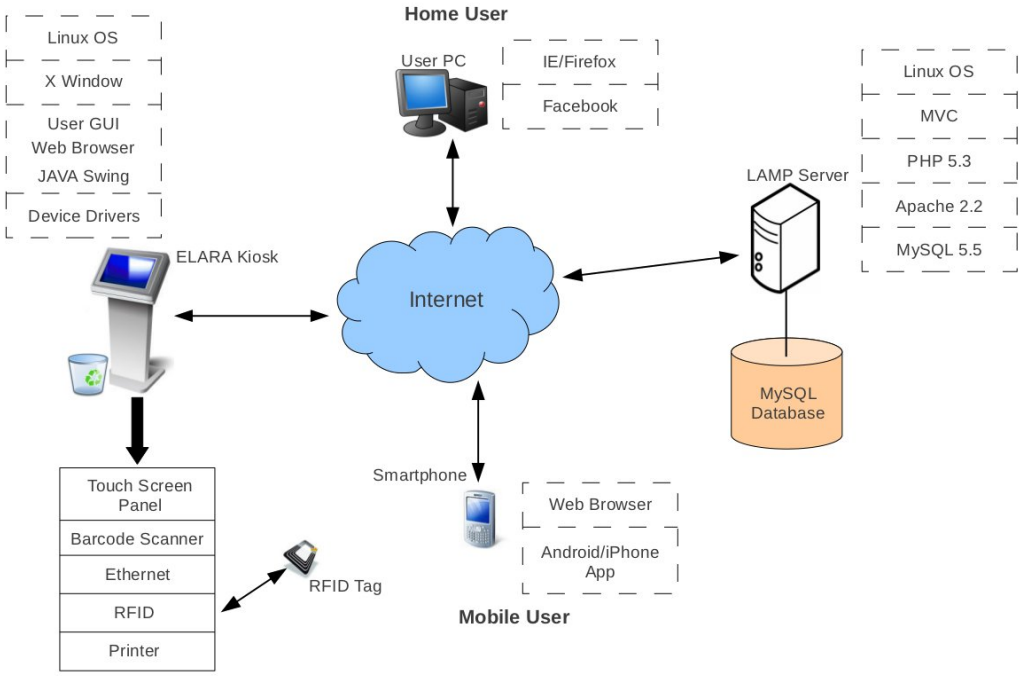

Figure 4.1: Overview of the complete ELARA system.

4.1 ELARA Kiosk: Touch Panel Computer

One of the most important components of the kiosk is the touch panel computer, which presents the user interface. Touchscreen devices have become very commonplace; the reason is that "their technology underpins attractive and responsive interfaces that are easy to modify for additional functionality" [10]. This is why I chose to use a touch panel for my kiosk to follow the current trend.

The touchscreen device that the ELARA kiosk prototype uses is the TS-TPC-8900, a complete touch panel computer manufactured by Technologic Systems [12]. The computer is comprised of an 800x600 10-inch LCD screen, which means that the diagonal of the screen is 10 inches across. The processor running the computer is the TS-4800 ARM Cortex-A8 CPU, which does not require a fan for cooling [12]. The TS-TCP-8900 shares many features with a laptop including two Ethernet ports, two

USB host ports, a headphone jack and a small speaker [12].

The TS-TCP-8900 has a pre-installed version of the standard Debian Linux distribution [12]. One particular feature of this particular machine is that the file system on the touch panel is "un-brickable", which simply means that it cannot be changed or corrupted without recovery [12]. Instead what happens is that the system can be recovered from an SD card in case the file system becomes damaged [12]. This "un-brickable" design became rather impractical when I was trying to incorporate my kiosk software into the touch panel computer because it would not let me add or delete any files and folders. Thus, the file system I use now is loaded through a USB flash drive which can be read and written. Being able to make modifications to the system was most helpful as I was able to begin testing my software application on the touch panel computer instead of a regular computer.

Since it is rather difficult to have the touch panel computer control low-level peripheral devices such as barcode scanners and card swipe readers, but it supports USB connections, the touch panel computer is connected to an Arduino microcontroller board, which manages all the peripheral devices. The Arduino microcontroller board plays an important role in the kiosk prototype; I will elaborate more on Arduino later in this chapter.

4.2 ELARA Kiosk: Touch Panel Software

4.2.1 Preliminary Work

As I needed to figure out how to get many people involved in generating information about recyclables and contributing this knowledge to a database, I conducted a literature search looking for papers on Crowdsourcing. After I decided on what it was that the crowd is to contribute to the database design and implementation, the next step was to design and build the database to hold all the information needed for the kiosk to be able to act as a recycling assistant. Once I had completed the database,

I proceeded with designing and creating a web-based prototype of the kiosk touch panel software. This allowed me to begin testing the database, while also beginning to establish a process for the user interactions with the kiosk, and helping me make some fundamental user interface (UI) design decisions. In order to make the overall system design, I used the traditional methodology of creating various UML diagrams (see below); at this stage, I also made a system requirements specification (see Appendix A). The last step of preliminary work was to create the first version of the kiosk touch panel software written in Java. The remaining time was used to connect this with the Arduino microcontroller and the database.

ELARA Client-Server Communications Prototcol

In order to make the communication between the client application on the kiosk and the server uniform, I created a communications protocol specific for ELARA leveraging the HTTP (Hyper-Text Transport Protocol). HTTP is a request/reply communications protocol. Request/reply protocols "guarantee that every message sent by one side is received by the other side and that only one copy of each message is delivered" [9]. Such protocols can also be "designed to protect the privacy and integrity of the data the flows over it, so that unauthorized parties cannot read or modify the data being exchanged between the client and the server" [9].

The ELARA Client-Server Communications Protocol defines the following rules:

- The touch panel sends an HTTP request with some data to the server.
- The appropriate server script processes the received data.
- Depending on the result that the script produces, it sends a specific numerical server response code indicating the result back to the client application on the touch panel.

The web-based software prototype is designed to follow this protocol, and the initial kiosk software is based upon the web-based implementation.

I chose to leverage HTTP since it is the protocol used by clients to communicate with servers across the Internet. This is particularly appropriate for ELARA since each kiosk must communicate with a remote web server containing the database with the required information. Thus, the ELARA touch panel kiosk software must send HTTP requests to communicate with the web server. I shall elaborate more on sending HTTP requests using the web client scripting language Javascript and the Java programming language later.

Network Security Measures

In kiosk systems such as ELARA, there is also an underlying level of security that the user does not perceive. Because user information is being sent across a network, it is crucial to establish security measures to prevent a malicious user from exploiting the kiosk. The possible security breaches I identified, which can occur at my kiosk, are Man-in-the-Middle attacks and Replay attacks. In a Man-in-the-Middle Attack, a malicious user intercepts all messages going between two parties, say Alice and Bob, and the attacker injects new ones, each time making them believe that they are talking directly to each other when in fact the attacker is controlling the conversation [17]. A replay attack is performed by a malicious user who retransmits a copy of message that was previously sent between Alice and Bob, making it appear to one of the parties, say Bob, that Alice is requesting the same thing again [9]. Although it was "not the original incarnation of the message", Bob would still consider it a valid message since it was created by Alice and it was not modified; then the attacker can potentially harm Alice or gain valuable information from Bob [9].

Since security breaches need to be prevented, I researched ways to prevent malicious users to take advantage of my system. First and foremost, it is important that the server can verify that the messages it is receiving are truly coming from an ELARA kiosk, and each kiosk must be able to confirm that the messages it receives from the network are truly coming from the server. To verify the authenticity of a

kiosk and the server, I determined that each kiosk be identified via a unique digital signature and the MAC address of the machine it is running on. These two unique identifiers are known to both the server and the kiosk, so if the messages include these two indentifiers, then authenticity of the sender can be confirmed. This method is useful for preventing Man-in-the-Middle attacks.

At the same time, replay attacks are to be prevented by adding a timestamp to each message sent back and forth between the kiosk and the server. These security measures are part of the communication protocol between a kiosk and the server, and the man-in-the-middle attack prevention measures are already included in the web-based version of the software; the present version of the kiosk software application includes both man-in-the-middle and replay attack prevention.

However, this is not enough. In an insecure channel, the malicious user can still see all the information being sent between Alice and Bob, so even if the server and the kiosk are including unique identifiers in the message, the attacker will be able to replicate the messages. There are two solutions to this issue: (1) encrypting the messages, and (2) using cryptographic hashes. I chose the second solution. A cryptographic hash is the result of a block of data undergoing a deterministic procedure that returns a fixed-length bit string called the message digest [15]. With a good hash algorithm, any change made to the block of data will result in a different hash [15]. Cryptographic hashing has many advantages; the main properties that I am interested in are the infeasibility to reverse engineer a given crytographic hash and the impossibility of having two different messages with the same hash [15]. Thus, this method is helpful in preventing both kinds of attacks. In particular, it is very easy to prevent replay attacks once a timestamp is included in the original message to be hashed since time is constantly going by. I shall return to the use of cryptographic hashing in ELARA when discussing the initial kiosk touch panel software application.

Web-Based Software Prototype

In the first major step towards the kiosk software application, I created a web-based kiosk software prototype. I established a user interaction process that allows for guest users as well as registered users who use my system. The reason for having registered users is two-fold; on the one hand, they can create a community of recycling advocates, and possibly contributors to the project database, and on the other hand, they form a good source for recycling data collection, which can then be used by researchers from various fields to examine recycling habits and trends in a particular location. The kiosk software therefore records whether or not a user recycled her item, and the user earns one point for each item she recycles.

The user interaction process I established takes four quick steps:

1. User sign-in,
2. Waste item identification through its barcode,
3. Question about correct bin for the item,
4. Deposition of item, and continuation or finish.

Once a user signs in in step 1, a new session for this current user is opened. Steps 2 through 4 constitute one transaction cycle since these steps can be repeated as long as the current signed in user wishes to use the kiosk for more waste items. When the user indicates that she is finished, or cancels her session in the middle of a transaction, she is logged off automatically, and needs to log back in if she wishes to do another transaction.

Version 1.0 of the web-based software only allows registered users to enter a barcode in a text field. The transaction is done all at once, so there is only one user/machine authentication process done at the same time. The recycleability of an item is determined as well. In order to communicate with the server, the web-based kiosk application sends an HTTP request using an XML Javascript XMLHttpRequest Object, which allows for data to be exchanged with a server behind the scenes [14].

Since this version of the software is only a prototype, the digital signature of the machine and its MAC address were hardcoded into the HTML code of the webpage in hidden input fields. I will not go into detail about the PHP scripts as these shall be further explained in section 4.5. The script for step 1 of the user interaction process authenticates the kiosk and the user - whether guest or registered, the script for step 2 authenticates the kiosk again and verifies the entered item barcode, and the purpose of the third PHP script is to record the transaction in the database.

While this version was a good start, it did not reflect an actual scenario at a kiosk since we cannot assume that only registered users of the project will want to use the kiosk. Therefore, I added the capability to allow guests users to use the kiosk as well. To make this prototype closer to the actual software, more functionality needed to be added to the prototype such as user authentication before entering the barcode, asking for a guess of which bin to put the scanned item in, asking ask for continuation or finished. I wrote a script that would simply authenticate the user before they could perform any further actions. Once the user has been authenticated, the prototype lets them proceed and enter a barcode into the designated text field. The user will only be able to submit a barcode if he/she has been authenticated. This concluded the development of version 1.0 of the web-based prototype.

At this point, I realized that the prototype software should be implemented via a Finite State Machine (FSM) reflecting each state of the transaction process with a single user. A FSM is "a mathematical model used to design computer programs" amongst other uses; it is an "abstract machine that can be in one of a finite number of states" [16]. Particular events and conditions cause the machine to transition from its current state to the next [16].

I began to incorporate FSM structure into version 2.0. With the first two versions of the web-based kiosk software prototype in place, I created some PHP scripts to stress test the database and the prototype to make sure they could process high query traffic and also to test the capacity of the database.

With the third version of the web-based kiosk software prototype, I modified the communication protocol between the kiosk and the server such that the prototype is in charge of dealing with the server response. I continued adding different state functions for the states of the finite state machine in the prototype. All four states were now implemented: (0) the initial state of the prototype before a transaction has been started (all forms on the page are blank and all global variables hold null), (1) the post-authentication state after the user has been authenticated, and the user can enter the barcode, (2) the question state when the user has to answer a question about the item they scanned, and (3) the continuation state when the user is given the option to continue with another transaction. The user can always switch back to the initial state during any other state of the FSM.

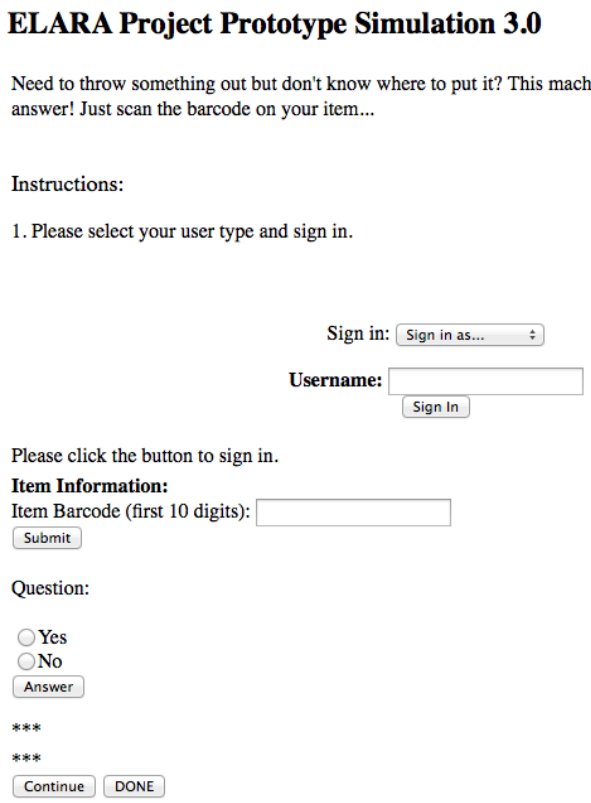

Figure 4.2: Screenshot of web-based kiosk software prototype

After a total of three versions of the web-based prototype, it was time to begin

with the initial touch panel software for the kiosk. This still used text input instead of barcode scanning or card swipes. The UI of the web-based kiosk software displayed all of these functionalities on a single webpage loaded by a PHP script. Fig. 4.2 shows a screenshot of the latest version of the web-based kiosk software. Instructions for the user are always displayed on the top of the page and change according to the current step in the session. The user signs in by selecting the user type in the dropdown selection menu, entering a user name into the text box if need be, and clicking the "Sign in" button.

Once the user is logged in, she may enter the barcode of her item into the appropriate text box and then submit this value by clicking the "Submit" button. In the next step of the process the kiosk asks the user if she thinks her item is recyclable or not, or in the case of an unknown waste item, whether or not the user will recycle. After clicking the appropriate button, the kiosk displays if the item really is recyclable or not and the user's current recycling rate, i.e., the total number of items she has recycled until that point in time. The user can then either click "Continue" to continue her session and start another transaction or "Done" to end her session.

Initial Kiosk Touch Panel Software

At this stage of the project it was time to implement the touch panel kiosk software design in the Java programming language as a networked client application which implements the finite state machine as established in the web-based prototype. I needed to develop a GUI program in Java to run on top of the client to create a much more user-friendly interface to the kiosk. (see `GUI.java` source code).

In this initial kiosk touch panel software, all the user input and application output happens through the shell. I did this for simplicity in testing. This touch panel software implements the four-state FSM of the web-based prototype walking the user through sign-in, item identification, bin indication and continuation of the

session with another transaction. In each case, the user simply enters text into the shell and the program processes the input. For each state of the FSM, the user enters information, the application creates an HTTP request with the user input and other internal data not visible to the user. All of this data is sent to the server which runs one of the previously written PHP scripts on the data depending on the current state. The application then waits for a response from the server containing other information, and then handles this response according to the current state of the application.

Another major component of the initial kiosk touch panel software was security. As described above, each message going between the client and the server must include a timestamp and the digital signature of the kiosk (the MAC address of the kiosk was no longer necessary). Since these two elements in the message are not enough to prevent attacks on the client-server communication, I chose to use cryptographic hashing to make all messages going between kiosks and server secure. The way it was done in the touch panel software was by taking all the contents of a given message, say the unique kiosk ID number and the username of the current user, but also a timestamp and the digital signature of the kiosk, and to apply a cryptographic hash function to this message to create a message digest. This digest is to be sent along with the message. However, the kiosk's digital signature is not included in the message, so it becomes impossible for an adversary to duplicate the digest because he does not have the signature. Without it, no digest he tries to create out of the plain text of the message should equal the digest. This ensures data integrity and also authenticity of the sender. Since both sides know the digital signature, once they receive a new message all they need to do is append this signature to the end of the plain message and run the same cryptographic hash function over the message, and compare the received digest with the one they just computed themselves. If they are identical, then the message is processed, if not, then the message is disregarded.

The specific cryptographic hash function I apply to all the messages is SHA-

512, which creates a 512-bit message digest, regardless of the length of the message. I use the Java MessageDigest class in order to create the digest and then convert the digest into a hexadecimal string representation of its contents with the `createDigest()` function (see `Controller.java` source code). The PHP scripts also generate hexadecimal string representations of SHA-512 message digests, which allows for direct comparison of digests.

Once I had the initial kiosk touch panel software working, it was time to build the GUI. I initially ported all the print statements to a separate GUI class, but the biggest issue was how to keep state between the two classes and how to establish communication between the two Java classes. Therefore, I incrementally moved away from the "all-in-one" Java class implementing the FSM, taking all its features and adding a layer of granularity to establish a clear data and control flow for the application (see Fig. 4.3). The initial version of the the kiosk software was the basis for the FSM class and the Controller class since it already dealt with keeping state, managing user input and changing the view accordingly.

4.2.2 Touch Panel Software Application

This application is written in the Java programming language Standard Edition 7. The entire application is comprised of six Java classes that all work together to create the kiosk software in its entirety following the Model View Controller (MVC) software architecture:

- The User class (User.java)
- The Serial Communication class (SerialCom.java)
- The Graphical User Interface (GUI) class (GUI.java)
- The Finite State Machine (FSM) class (FSM.java)
- The Status class (Status.java)
- The Controller class (Controller.java)

Fig. 4.3 shows the relationships between these classes.

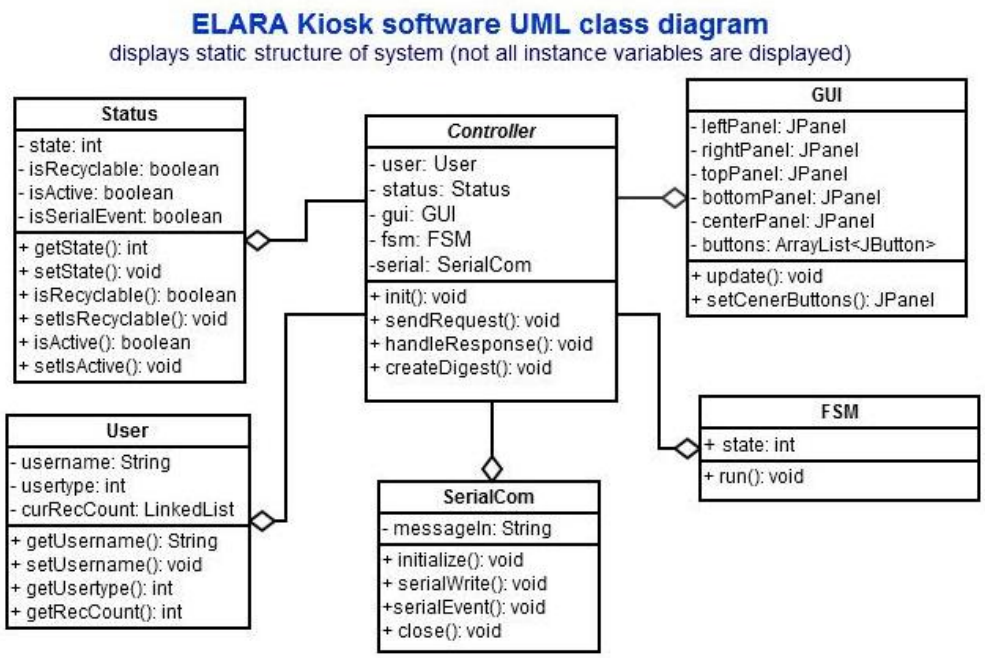

Figure 4.3: UML class diagram for the kiosk touch panel software

Model View Controller

MVC is a common design pattern used in Software Engineering. Its main purpose is to isolate the different aspects of a software application (data, control flow, presentation) while maintaining a flexible coupling of each aspect [3].

More specifically, the model aspect of the software application implements the logic for the application's data domain; for example, the model is often linked with a database which holds all the business data [3]. In ELARA, the User class is part of the model.

The view aspect of the application is the component which creates the user interface, i.e. it displays the data of the model. In the case of ELARA, the GUI class represents the view.

The controller aspect of the software is what truly glues together the model and the view. It manages user input, communicates requests for data from the model component, and determines which view to show according to the user input. This

means the controller triggers a change of state as a result of the user input. The controller in ELARA is represented by the FSM, the Status, the Serial Communication, and the Controller classes.

The Model

The User class defines the most basic properties of an ELARA User. Getter and setter methods for the four variables are also defined (see `User.java` source code). The data itself is kept in the database on the system server.

It is necessary to separate out the User class because ELARA users, especially in the future, will have many properties which are used to personalize the user's experience at the kiosk. Currently, the User.java class only defines the most basic and necessary properties that all users, both guest and registered users have in common. Future work will be to incorporate a Registered User class which will have all the properties specific to an ELARA registered user.

The View

The GUI class is the only component of the kiosk touch panel software which is part of the view aspect. This class manages the appearance of the program at any given time and for any possible state of the FSM, primarily by using the `javax.swing.*` and the `java.awt.*` packages. These packages provide the graphics platform for the UI, and enable events, such as button presses, to occur on the graphics context, allowing the user to get a comprehensive presentation of the data, and giving her the means to communicate with the software through the touch panel computer. Java uses so-called panels which serve as a canvas of sorts to display certain content.

The overall layout of the GUI is another element that is set at the very beginning and then does not change apart from its contents. The layout of the main panel of the GUI is a `BorderLayout`, which is partioned into five sections:

Fig. 4.4 shows how the screen is partioned using this layout; it is divided into a

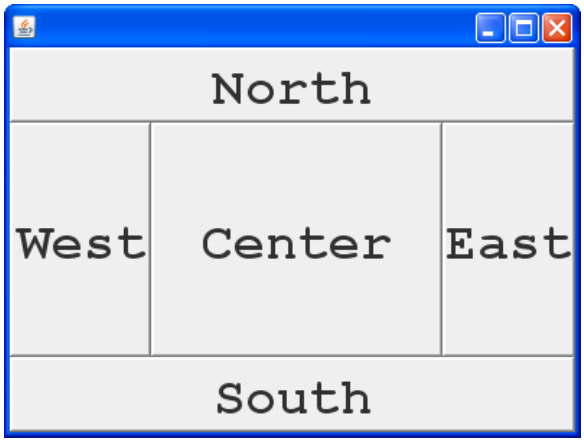

Figure 4.4: BorderLayout JPanel partitioning. Image courtesy of www.java2s.com

North, a South, an East, a West, and a Center section. ELARA only uses the North, Center and South sections so all the contents of the panel are centered on the screen. I use various helper functions inside of my GUI class to create smaller panels for each section of the layout, so that I can also insert contents into each small inner panel if need be (see `GUI.java` source code).

There are two main functions inside the GUI class that play a major role in how the display works and how the user can interact with the kiosk. The first function is `setCenterButtons()` which manages the center buttons to be displayed in the Center section of the main panel.

The other major, and most important, function in the GUI class is `update()`. This function manages the entire display for every single state of the touch panel software FSM. This function also needs to arrange for the correct buttons to be displayed (if any) at any given time. Therefore, it calls the `setCenterButtons()` function in states that require the user to press a button on the screen.

`update()` is called by the Controller class at the beginning of each cycle (when the FSM might have changed to a different state) to update the screen according to the current state.

The Controller

The controller aspect of the kiosk touch panel software is split into four classes: FSM.java, SerialCom.java, Status.java, and Controller.java. These four classes together are the "brain" of the touch panel making the kiosk operate as designed.

The Finite State Machine class is based on the FSM in the web-based software prototype as well as the one in the intial kiosk touch panel software. However, the four intially proposed states are not enough to handle the entire user interaction process in a practical way. Therefore, the user interaction process was further divided into smaller tasks and the number of states at present is 24. These states are defined as integer constants in the FSM class. Each state is attributed to a small task that needs to be done during a transaction; for example, in the `MAIN_MENU_STATE`, the GUI will display the main menu asking the user to choose their next action on the kiosk. At present, the only action the user will be able to take is to start a transaction, but more options are possible in the future. Many states remain unused at the moment since their respective parts in the user interaction process have not been implemented in the Controller class yet. However, I have included them for future work purposes.

While this class does not define any instance variables (and thus getters and setters), it defines the state transition function of the ELARA FSM. The state diagram for the kiosk touch panel software is shown in Fig. 4.5.

The `run()` function inside of the FSM class implements the state transition function; it checks the current state, and then changes to the next state depending on the values of certain other variables in the Status class. The conditions for every other implemented state of the FSM look similar (see `FSM.java` source code).

The serial communication class manages the data and control flow between the touch panel computer and the Arduino microcontroller. This class is in charge of opening a serial connection (USB) between the touch panel and the Arduino. Data must flow both ways since the touch panel must be able to send commands to the Arduino to operate certain sensors and effectors, while the Arduino must be able to

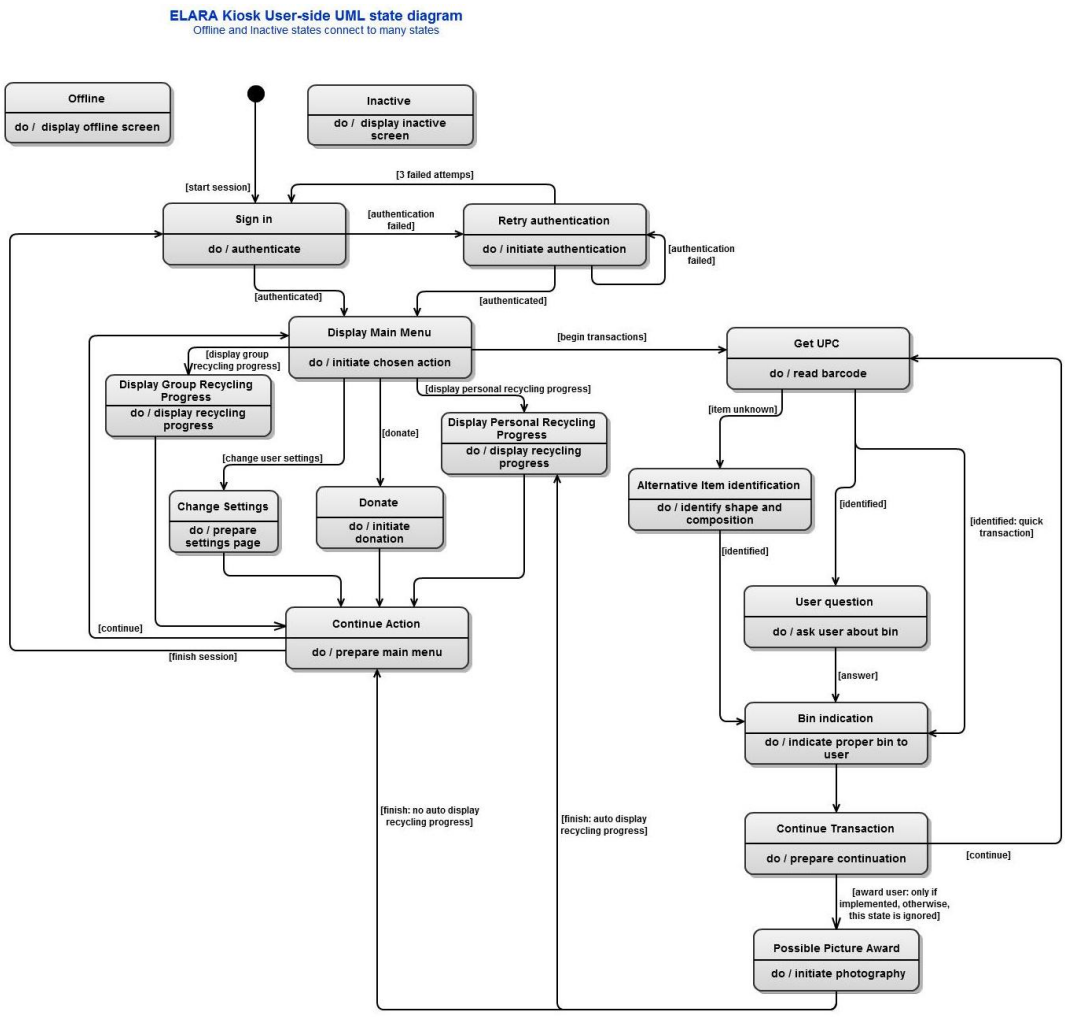

Figure 4.5: UML state diagram for finite state machine of the kiosk touch panel software

send data from the barcode scanner, card swipe reader and sensors.

This class defines four major functions:

The `serialWrite()` function manages the data flow from the touch panel to the Arduino. It takes the message to be sent to the Arduino as a parameter, and then writes this string out character by character to the OutputStream of the serial connection.

The `serialEvent()` function is called when the SerialCom class detects data coming from the Arduino. The other two functions that the SerialCom class defines

are `initialize()` and `close()`, which initialize and open an serial port between the touch panel computer and the Arduino, and close this port, respectively.

The next very important class which is part of the controller of the touch panel software is the Status class. This class contains all the shared data between all the components of the kiosk software. The idea to create this class did not come until after I began experiencing difficulties sharing data between the FSM and the GUI while working on the initial kiosk touch panel software. The main purpose of this class is to keep the current status of all the components of the application and that all components of the touch panel software can access this data, most of which is required for the Controller class and FSM to operate properly.

One issue that I faced while testing the software on a regular computer platform, was that many of the variables were not being set when I was expecting them to be set. After going through some debugging procedures, I realized that the issue arose from lack of synchronization of the threads. Each program is run by a thread that the computer executes as a single unit. The issue with GUI programming is that GUIs in Java run a second thread causing race conditions. Thus, the GUI must race against the Controller to write and read the Status variables. The solution to this is to make all functions and variables of the Status class synchronized. What this means is that it is ensured that a single thread has access to the function or variable and is allowed to finish its execution on it before the other thread is allowed to access it. This prevents variables not being set with unexpected values.

The FSM and Status classes are key to the proper functioning of the kiosk touch panel software, but the Controller class is the most important part of the controller aspect of the software. This class takes care of keeping the kiosk software running, and communicating with the server on the user's and Arduino's behalf. The functions in this class are largely based on those I wrote for the initial kiosk touch panel software, but have been modified slightly. This class defines mainly constants that are required to communicate with the server, such as the names of the specific

server scripts that are called, and the server response codes (see `Controller.java` source code). These codes are unique to each type of response my server script produce and constitute the communications protocol between the kiosks and the server.

Similarly to the GUI class, the Controller class also defines many auxiliary functions. The three major functions in the Controller class are `sendRequest()`, `handleResponse()`, and `createDigest()`. In the `sendRequest()` function, a connection to the server through a Socket is opened, and a HTTP request is sent. These two functions work in conjunction and control the kiosk-server communication.

The Controller class creates the connections to all the other Java classes, and then contains an infinite loop which will run the GUI `update()` function, perform a specific action according to the current state of the FSM, and will run the FSM `run()` function to transition to a new state after this action has been performed.

Since this software is only a proof-of-concept version, many features can still be added to enhance the kiosk (see Chapter 5). I plan on adding some of these features in the near future myself, others are left for other students to work on in the future.

4.3 ELARA Kiosk: Arduino Microcontroller Board

The second major component of the ELARA kiosk is the Arduino microcontroller board, which is an open-source electronics prototyping platform [2]. A microcontroller is a small computer on a single microchip containing a processor, memory, and input/output ports [18]. Microcontrollers are designed for embedded devices such as implantable medical devices, remote controls, appliances, and car multimedia systems [18]. I was advised to use the Arduino since it is flexible, easy-to-use, and "intended for anyone interested in creating interactive objects or environments" [2]. This added flexibility to control devices that connect to the Arduino board is what allowed me to design and create the interactive kiosk that I wanted. Due to time constraints and lack of expertise, I mainly assisted with this part of my project.

I am using an Arduino Duemilanove board (see Fig. 4.6), which has the following features [2]:

- 14 digital input/output pins,
- 6 analog input pins,
- one USB connector.

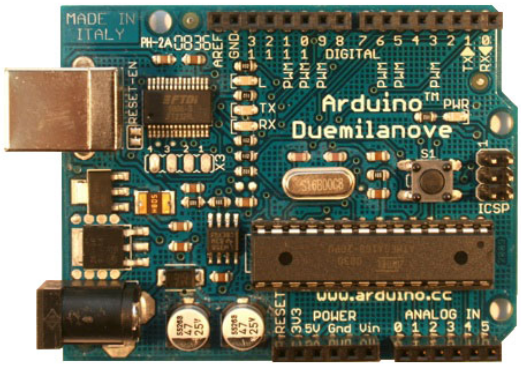

Figure 4.6: Arduino Duemilanove board. Image courtesy of www.arduino.cc

Since we needed to connect a barcode scanner, a card swipe, two LED strips, and three Sharp sensors to the Arduino, we also needed an Arduino ProtoShield, which allowed us to build a customized circuit for driving the LEDs and convenient connectors. Thus, the ProtoShield holds nine pins for the three Sharp sensors, eight pins for the LED strips and three pins for the PS/2 connection of the barcode scanner and card swipe. In addition, we also added several high current transistors to drive the LEDs. The entire Arduino board with all the devices connected to the Arduino is displayed in Fig. 4.7.

One key feature of the Arduino Duemilanove is the fact that it has three different kinds of memory: Flash memory, which contains the bootloader (a program that sets up the hardware and software when the device is first powered on) and the instructions of the executing Arduino program, the SRAM, a form of volatile memory,

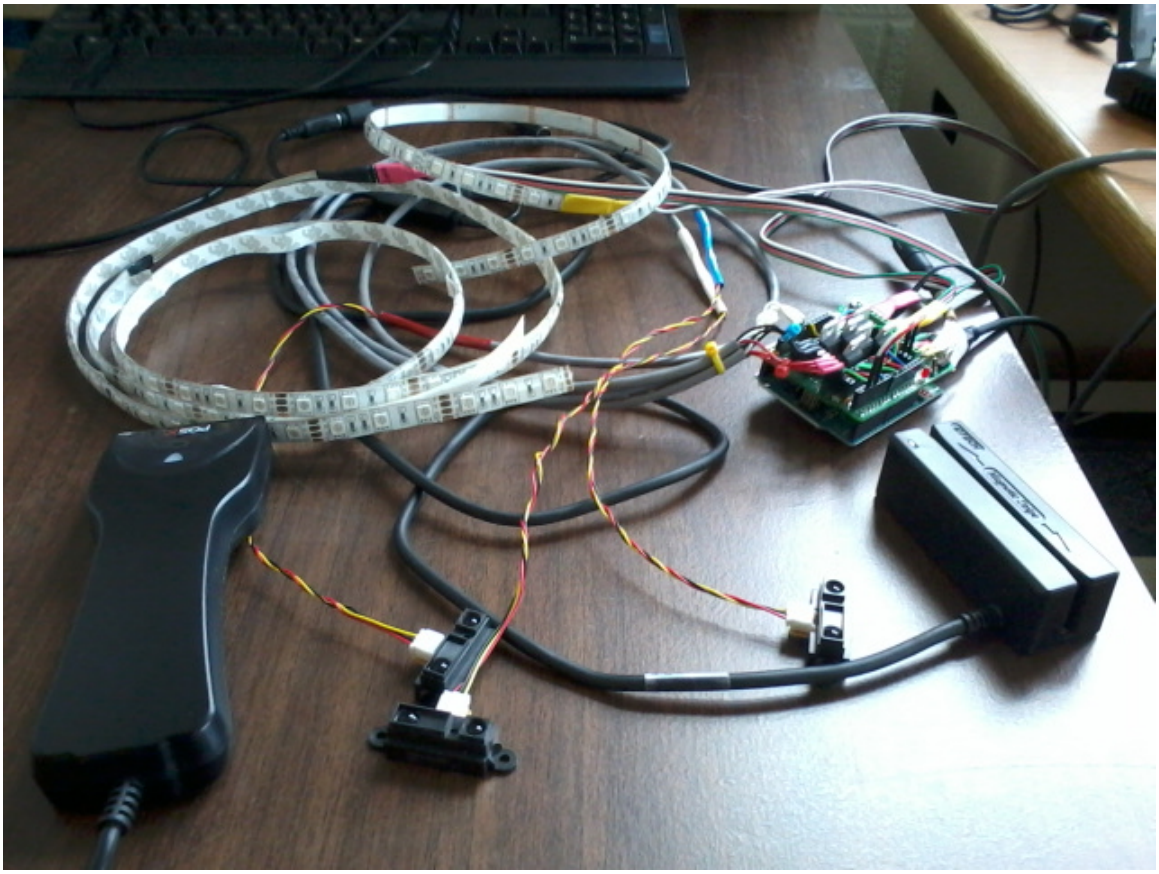

Figure 4.7: Arduino Duemilanove board and ProtoShield with all peripheral devices connected

and EEPROM, which serves as non-volatile storage for the Arduino. The EEPROM is especially important as it stores information about each of the devices that it connects to; this information is needed for the Arduino software to communicate properly with the touch panel software.

4.4 ELARA Kiosk: Arduino Software

The Arduino Duemilanove can be programmed using the Arduino programming language, a version of C++ [2]. Similar to the touch panel software, the Arduino software can be divided into two groups cover two main tasks:

1. Programs that control peripheral devices,

2. Programs that manage Arduino to touch panel communication.

The functions of the first group of Arduino programs are rather straightforward. We wrote three small programs that manage the LED strips, the Sharp IR sensors, and the EEPROM (see source code for `led.pde`, `sensors.pde`, and `eeprom.pde`).

The LED program defines the driver code for two RGB LED strips connected to the Arduino. Each LED strip must be connected to four pins, one for the power source, and three pins for the color values that the strips are set to. Each color value (one for R, one for G, and one for B) needs to be between 0 and 255, and a combination of all three RGB values results in a particular color. At present, the LED Arduino program defines four functions, one to intialize the LED strips, one to turn both strips off, one that cycles through all colors in the rainbow, and one that turns a specified LED strip on to the specified color. The Arduino calls the appropriate function after the touch panel software has sent a command for one of these functions during a transaction; this occurs at the beginning of a new session since the rainbow program is running when the kiosk is showing the welcome screen, and the program that turns on a specific LED strip on to the specified color is called when the kiosk indicates the proper bin for the user's waste item.

The Sharp sensor program defines the code area for reading break-beam or infra-red sensors, button presses, or any other user actions at the kiosk. The current design, however, only supports binary 0/1 sensor data. We use a fixed length string of characters ('0' or '1') to indicate which sensor is being read. The `readSensors()` function in this program goes through each defined sensor, and checks to see which one has received a signal. The purpose for these sensors is to indicate when the user has inserted her item into the bin, and they also serve as a check for which bin the waste item was inserted into. This function is called when the touch panel software sends a command to read the sensors at the stage in the transaction when the correct bin for her waste item is being indicated to the user.

The third program in the first group of Arduino code is the EEPROM control program. Its job is to read and update the EEPROM where peripheral device system parameters, and other properties and settings for these devices. A very important piece of information that is stored in EEPROM are the device point-of-sale device prefixes, i.e., the message prefix that the barcode scanner and card swipe prepend to each message they send to the Arduino. These prefixes let Arduino distinguish from which device a message is coming. Various functions for adding, deleting, editing and loading the information stored in EEPROM are used in the communication between the Arduino and the touch panel software.

As with the communications protocol I established between the server and the touch panel software in Section 4.2.2, we also established a messaging protocol to share messages between the Arduino and the touch panel software application as part of the second group of Arduino programs. We wrote a program that contains the definitions and functions to manage the message protocol; we have implemented a simple, parity checked protocol that passes peripheral device data as character strings as well as commands from the kiosk touch panel software, and responses from Arduino. Messages have the following format:

Preamble: @@

Command: A character from A-Z

Data: Varying number of characters based on the command

Terminator: The carriage return (Enter) character

Parity byte: Computed by XORing all previous message characters

Currently, while the function to compute the parity byte has been written, this byte is not sent along with the rest of the message for simplicity in testing. The other function in this program is the `createMsg()` function which creates a message with a specified command and specified data to be sent to the touch panel computer.

The fifth Arduino program is the one which creates the link between the Arduino and the kiosk touch panel. This program defines the commands for the mes-

saging protocol, and handles the data collection from the various peripherals. We have defined 13 possible commands that can be sent from the Arduino to the touch panel, in both directions, and from the touch panel to the Arduino. I consider this program the Arduino counterpart of the Controller class of the touch panel software application since it uses all the other programs and runs them.

In particular, it contains three functions that manage each aspect of the communication between the Arduino and the touch panel. The `posEvent()` function processes the data coming through the PS/2 connection of the barcode scanner and the card swipe. `serialEvent()` is called regularly by this program and processes new data coming in from the physical serial connection between the touch panel and the Arduino. Lastly, the `processKioskMsg()` function extracts the message contents coming from the touch panel, and determines the course of the action to take according to the received message.

Similar with the touch panel software, these Arduino programs have been designed to be expandable for future work purposes, but establish a complete Arduino-to-touch panel communication at present.

4.5 ELARA Server

While the ELARA kiosk is the most immediately noticeable component of the ELARA system, the server plays a central role for all of the components of the entire system. As described in some detail above, the touch panel software application needs three server-side scripts that allow data exchange between the kiosk and the database. I wrote these server scripts in PHP during the preliminary stage of the touch panel software development (see Section 4.2.1). Each script serves requests for one of the major steps in the transaction process: `authentication3.php` manages user authentication, `transaction5.php` is used at the item identification step, and `record3.php` is in charge of recording each transaction in the database (see source code for these

three scripts).

As noted above, all three scripts always authenticate the requesting kiosk first by checking that the kiosk ID number sent in the message is on record in the database, the timestamp on the message is no more than five minutes old and by performing the message digest check. In addition to these three security tests, all three PHP scripts also check that the kiosk's current number of transactions is accurate since a repeated number indicate that an error occurred either on the client side or in the network. If the sender cannot be recognized at some point and the message in fact does not pass one of these four tests, the script will generate a response message with the appropriate server response code indicating the error that either the machine was not recognized or that the message was rejected. If the kiosk message has passed these four tests, the scripts query the database for the appropriate data, and then generate the appropriate response.

In the case of the user authentication script, the PHP script will query the database for the username contained in the received message. The script will generate its response either with the server response code indicating that the user could not be found in the database, or with the user's unique ID number.

The `transaction5.php` script will query the database for the item barcode contained in the received message. If the item is not on record, this script will generate a response message indicating that the item waste type is unknown. If the item is on record, the script will check to see if the item is recyclable, landfill waste, or compostable. I wrote a function which determines whether an item is recyclable or not based upon how many votes it has for being recyclable and how many votes it has for being landfill waste. The larger the number of votes, the higher the percentage of votes for being recyclable the item must have to truly qualify as being recyclable. it is also possible for the item to have an unknown waste type if the number of votes for both recyclable and landfill are around 50% each. I have determined four ranges of number of votes to implement this quality check on the crowdsourced information.

This function returns the server response code indicating whether or not the waste item was recyclable, landfill waste or unknown. However, this function does not take votes for being compostable into consideration. I plan to add this feature in the future as an additional feature; at the moment, the script determines that an item is compostable if it is not recyclable but has at least one vote for being compostable. Once the waste item type has been determined, the script generates its reponse with the appropriate server response code indicating the waste item type, or that it is unknown. The server response codes of this script are used by the Controller to indicate the appropriate bin to the user to dispense her item if the waste item type is known. If the waste item type is unknown, the controller will prompt the user to an alternative waste item identification process; this process is also to be implemented as an additional feature in the future.

The server script which records the transaction, and which is requested after the user has dispensed her item in one of the bins of the kiosk, will query the database for the current user's total number of items recycled which shows her recycling progress, and record a new transaction in the database. In the case that the current user's last waste item was reyclable and the user inserted it into the correct bin, the user's recycling progress is increased by one. This number is sent along in the response message as data. If the user is a guest, the script will not return a recycling progress. If the transaction is recorded successfully, the script will indicate this with the appropriate server response code in its response message.

Lastly, a few words on the generated server response message by these three scripts are necessary. Each response message also contains a timestamp and a message digest so that the touch panel software can also verify the sender of all the messages it receives.

4.5.1 Waste Items Database

The ELARA database is crucial to the system since it contains all the data necessary to make the kiosk function as a recycling assistant. This component of the ELARA system was the first one to be finalized since the touch panel software coud not be designed properly and operate without being able to query the final database. I chose to create a MySQL 5.0 database with nine tables. At present, only four of those tables are in use for the proof-of-concept version of the ELARA kiosk; the other five tables provide the necessary space for expansions of the project. Fig. 4.8 shows the entity-relationship diagram of the ELARA database.

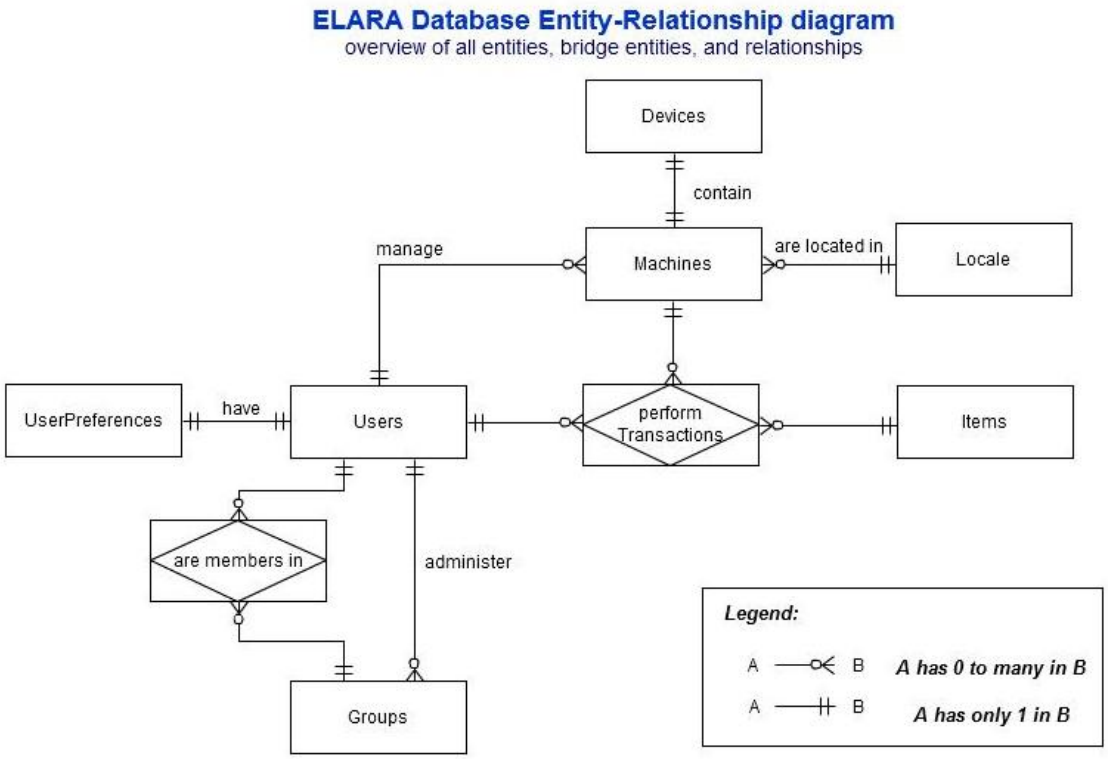

Figure 4.8: Entity-Relationship Diagram for the complete database

The four tables being used are the `Users`, the `Machines`, the `Items` and the `Transactions` tables. The attributes of the `Users` table are for information pertaining to a single registered user's geographic location, potential affiliations to institutions,

personal data such as occupation and age, and also information pertaining to her recycling profile. At the same time this data is used as part of the user's profile on the ELARA website. This table is linked to the `User Preferences` table which contains many possible profile settings that can be seen and set either on the website or at the kiosk; however, the functionalities from this table remain unimplemented. The `Machines` table contains the information about all the registered kiosks. This information includes the digital signature of the kiosk, its location, whether or not it is active and how much has been recycled at that kiosk. This table is linked two other tables, the `Locales` table which mainly describes recycling policies of all kiosk locations on record, and the `Devices` table which holds information about specific devices that a particular machine has as part of its kiosk. For example, it indicates how many bins a given kiosk has and the purpose of those bins. These two tables are also currently not in use.

The `Items` table contains all the data related to all waste items on record; each item has a barcode, an associated waste type that users vote on via the ELARA website, and also more specific information about the item such as its material, its shape and its size.

The `Transactions` table brings the `Users`, the `Machines`, and the `Items` tables together. Every time a new transaction is recorded, the user ID or the current user, the kiosk ID of the kiosk involved, and the item ID of the current waste item are recorded as well as whether or not the item was recycled and at what time this transaction took place. The data in this table can be used to generate statistics and graphs about recycling trends in specific locations, or by specific items in a given time frame.

There are two more tables that remain unused, namely `Groups` and `Group_Members`. These two tables have been built to provide space for expansion of the ELARA user community and to allow user groups to form. I shall elaborate more on this topic in Chapter 5.

4.6 The ELARA Website

In the previous section, I described how much of the information in the ELARA database is also needed for the ELARA website to function. The website has four major functionalities:

1. A site where any visitor can find information about the project,
2. A platform to display the recycling progress at participating locations, and to show recycling trends,
3. A space for registered ELARA users to see their personal recycling profile, and to interact with other ELARA users,
4. A crowdsourcing system to collect knowledge about recycling and waste.

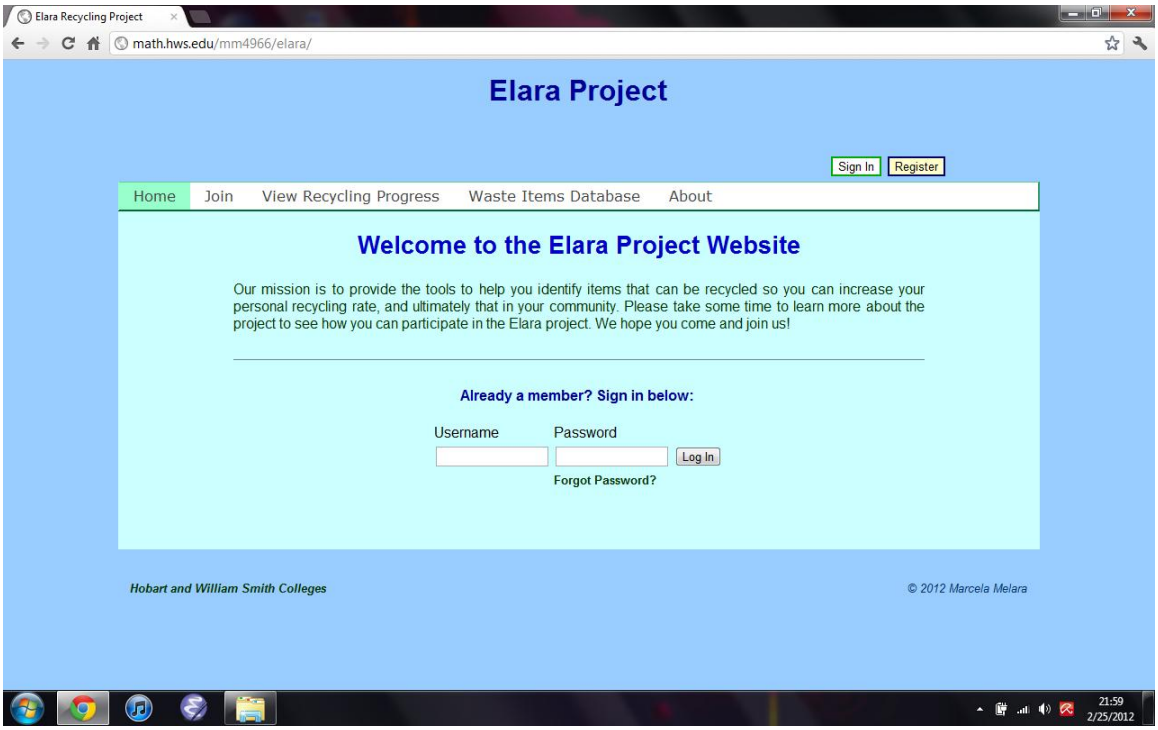

Figure 4.9: Screenshot of the ELARA website homepage.

In general, the ELARA website has several public pages which explain what the project is about, and which display various aspects of the crowdsourcing system. One page is dedicated for any user to contribute to the project either by adding a

new item to the database, or by voting on the item being recyclable, landfill garbage or compost. The website is also a platform for displaying the recycling progress at participating locations and to show recycling trends at this location. The main purpose for these pages is for researchers in various fields to be able to analyze this data and use it in their research. All of these pages are publicly visible and do not require a username and password to access them.

The private pages of the ELARA website which do require user login are those displaying personal user profiles. In this realm of the website, any registered user can view their own profile, and also register a kiosk they are managing. These private pages also allow registered users to write messages on the user bulletin board to share their own stories, or to leave other important messages about their experiences.

At present, the ELARA website is only functioning as a public site where anybody can find general information about the project (see Fig. 4.9). The other four functionalities are to be implemented in the future.

5

Future Work

The complexity of this project has left much space for future enhancements and expansions in many aspects. This chapter describes only a select number possibilities that I find most compelling and viable in the current context of the project. The future work I describe does not pertain to the enlargement of the network of kiosks, but rather to added features and functionalities.

5.1 Additional Kiosk Features

In the previous chapter, I describe the current features of the ELARA kiosk. At present, since the kiosk I have built is a proof-of-concept model, I have decided to exclude many features that I have designed the database to support. While there are a myriad of ways to enhance the kiosk, there are four enhancements I find are most interesting and also necessary to keep up with modern technology trends:

1. The possibility to make donations to charities through the kiosk,
2. Instant user rewarding by taking pictures of record-breaking users,
3. Audio-enhanced user interface,
4. Image-recognition of barcodes or item labels.

The possibility to make donations can be a nice feature for an ELARA kiosk since it adds to the "feel-good" aspect of using ELARA. Users would be able to select a charity and an amount to donate via the touch screen, and they could then swipe their credit card at the kiosk to make the donation. Since ELARA kiosks already have a card swipe, this feature could be easily added from a hardware-oriented aspect. However, this would also require computer and network security enhancements to protect the users' credit card information. Nevertheless, because other service kiosks, which must also handle credit card information, have become very widespread, their security measures can be bootstrapped facilitating the addition of this feature.

Adding instant user rewarding capabilities can also be done rather easily. The purpose of this feature is mainly as an incentive for users to feel instantly recognized for their efforts. The way this capability would work is that the kiosk would notify the user that she has broken a new recycling record at that kiosk, for example she has recycled the 100th water bottle, and it would offer her the option to get her picture taken via an integrated camera. This picture would then be featured on the ELARA website naming her one of the champion recyclers of the day. Since the ELARA website is already being used as a platform for users to share their stories, adding a page for these champion recycler pictures is an easy addition to make. However, the kiosk software, both that of the touch panel and that of the Arduino, as well as the server scripts and the database require significant changes making this a more ambitious extra to add.

Another rather difficult feature to add is the audio-enhanced user interface. At present, only people who are not visually handicapped are able to use the ELARA kiosk because of its graphical user interface. Adding spoken instructions and voice-recognition functionalities to the kiosk would broaden the range of users who could use the kiosk.

Lastly, the image-recognition of barcodes and/or item labels would be another great feature for the kiosk to have since it takes away the need for a barcode scan-

ner which are often finnicky in reading barcodes and require prolonged exposure to the barcode to read them. Having a barcode-activated camera would facilitate item identification for the user. However, integrating this feature into my initial kiosk touch panel software was not possible in the time given. Nevertheless, this is another functionality that can be added to the kiosk to make it more user-friendly.

5.2 Competition Component

As discussed in Chapter 3, one major aspect of crowdsourcing are competitions. With a large enough user community, ELARA users could form user groups according to affiliations with institutions or in their municipalities, and have recycling competitions. This would be a great way to get more people involved in the project and help increase the recycling rates in their communities. These competitions could be organized by a particular group which could be challenging other groups. The advantage of letting the users themselves organize these competitions would be that they could set the time limits and the stakes of the competitions. The ELARA website could then provide them with the space to document the recycling competitions. Having the possibility for a competition component would make ELARA a true crowdsourcing system.

5.3 Smartphone Application

With the ubiquity of smartphones today, creating a smartphone application seems like a very logical step to take. When I first had this idea, I wanted to write this application as a portable kiosk but soon realized that this would not be possible with the security measures I have established, and it would be very difficult, if not impossible, to verify that the user truly discarded the waste item in the proper bin and was not cheating to increase her recycling score. Thus, the main functionality

of the ELARA smartphone app can be to simply indicate the waste item type of a particular waste item given its barcode or its shape using the same alternative item identification procedure as in the regular kiosk. This way, this app could be used anywhere regardless of whether or not there is an ELARA kiosk in the area. Related to this is another functionality that the ELARA smartphone app could have. Since it is location-independent, it could also be used to locate the nearest ELARA kiosk so that the user can actually record her transactions in the database.

The app does have one major disadvantage: since it does not record individual transactions, registered users who use the app do not get the credit for recycling items that they identified using the app. Nevertheless, the scope of the app could be so wide that it could still help many people recycle and increase the recycling rates whereever they go.

6

Discussion and Conclusions

6.1 Discussion

When I first started working on ELARA in the summer 2011, I had a large number of ideas that I wanted to implement. From the very beginning, I was planning on building a proof-of-concept networked kiosk, writing the software for it, and building a website at which people could contribute to the ELARA project. I was also planning to release my kiosk to the HWS community at the beginning of the spring semester.

I made much progress during the summer by doing the vast majority of the preliminary work to build the database, write the kiosk touch panel software and begin working on the ELARA website. I realized that my project was very ambitious in terms of finishing all the components I set myself out to work on in the allotted time, so I began to narrow down my focus on more specific elements of my project from a very early stage. Nevertheless, I found myself adding more tasks to my agenda that were necessary for improving my design, and also to ensure the proper functioning of my kiosk.

Once the fall semester began, I was expecting to continue with my project from where I had left off, but since I did not have my design completely finalized, I worked on creating various UML diagrams which included many possible future

expansions of the project; this would allow me to solidify my kiosk software class design, as well as the database design, and make these designs expandable without having to modify much about the software. In particular, I paid attention to possible hardware additions to the kiosk on top of the planned barcode scanner, card swipe reader, LED strips and Sharp sensors. Solid design requirements facilitated the process of writing the Java classes for the touch panel software, but it also enabled me to establish the System Requirements Specification, which describes all the conditions and requirements for building an ELARA kiosk and connecting it to the network (see Appendix A).

Having expandable Java classes for the kiosk touch panel proved very useful towards the end of writing the software, when I began to make decisions about what I truly needed the software to include for beta testing purposes. It allowed me to have a fully working kiosk touch panel even if not all the features I had originally planned for were included yet. At present, I have yet to include three elementary features in the kiosk touch panel software:

- Alternative item identification,
- A Help screen for each screen in the transaction process,
- Keeping a log file.

Alternative item identification will add significant power to my kiosk as users will not be bound to items with barcodes. The Help screen, while a rather unconspicuous necessity, is a much needed feature if the ELARA kiosk is to be considered a true service kiosk. Keeping log files for keeping track of any errors can be helpful from the kiosk management point of view since it will help other future kiosk managers and myself from figuring out what error occurred, where and when it occurred, and also fixing these errors.

Having written the final version of the server scripts that serve the kiosk requests during the preliminary stage of the project also allowed me to focus more on

other aspects such as the kiosk software and the hardware design and implementation, as it became vital to me to finish these two aspects first.

The ELARA website at present does not support any features for registered users, but is almost fully functioning for the general public. The only features I still need to add to the public part of the website are the public viewing of the recycling progress and the ability to add items to the database.

All in all, one of the main things I learned is that time management and planning ahead can go a long way in making significant progress. At certain points, I tended to underestimate the amount of time a certain task would take. Thus, this project was not only about learning to conduct substantial computer science research but also about learning to finesse time management and planning skills. Although many compromises had to be made due to time constraints, I would consider my project fairly complete. In the middle of the project, I felt I had not accomplished much, but looking back now, it has all come together to form a great proof-of-concept system.

6.2 Conclusions

ELARA is a system that combines several areas in computer science: embedded computing, software engineering, networking, databases, web development and human-computer interaction. This makes ELARA an enormously intricate hardware and software system. ELARA is a very ambitious project, and I still have many more ideas for potential extra features and directions this project could go. Working on ELARA for 12 months has been both stressful and frustrating, but this is far outweighed by the incredibly enlightening and rewarding experience. With this project, I tested my limits, and it has prepared me to conduct intensive research in graduate school, an experience I look forward to with even more enthusiasm now.

My hope is that ELARA will make an impact, even if only locally at HWS,

because this will have meant that all the energy and time I put into this project will have been completely worthwhile. Should something more grand result from this project, that will have been more than I had hoped for. The completion of this thesis and my graduation does not mark the completion of ELARA. My plan is to continue working on this project as far as time allows; the next step for me will be to begin working on a smartphone app for the Android platform as many people have been asking to see one.

In conclusion, the ELARA kiosk is a proof-of-concept device showing that my idea can be put into action in a practical manner. The kiosk software and the hardware design in their entirety are open-source so that any interested person can improve upon it, and/or use it to build their own kiosk and join the network. While the extent to which crowdsourcing will truly help my system has not been tested, the feedback I have received about the ELARA kiosk has been positive altogether.

Appendices

Appendix A

System Requirements Specification

for

ELARA: Environmental Liaison and Automated Recycling Assistant

Version 1.0

Prepared by Marcela Melara

Hobart and William Smith Colleges

21 September 2011

A.1 Introduction

A.1.1 Purpose

The purpose of this document is to describe the system requirements specifications for the kiosk subsystem of the ELARA project. More specifically, the contents of this document specify the required hardware and software features for the ELARA kiosk

(client) and the ELARA server.

A.1.2 Project Scope

The ELARA project is a computer enabled crowdsourcing project designed to first and foremost improve recycling rates at small institutions in the United States. Via a networked embedded system of kiosks all communicating with a central server, ELARA assists people with separating their landfill garbage from their recyclables in an educational yet entertaining manner. This project is designed to enhance the separation and the recycling experience at a personal and interactive level providing motivation to become more aware of their waste separation habits and thus to increase their personal recycling rate. At the same time, the project uses the wisdom of the crowds about recyclables to build the knowledge the automated recycling assistant uses to perform its task.

The software being specified in this document is (1) the application program running on an embedded computer at an ELARA kiosk, and (2) the scripts running on the server side of the subsystem. The kiosk application program provides the user interface with the system, i.e., it is the front-end of a larger system, and it is the piece of software which will assist the user in their recycling activity.

The hardware requirements being laid out in this document describe the constitution of an embedded computer at an ELARA kiosk, as well as the server.

A.1.3 Definitions, Abbreviations, and Acronyms

Term	*Definition*
ELARA	Name of the project; stands for "Environmental Liaison and Automated Recycling Assistant"
Item, waste item	Refers to any kind of waste item such as a recyclable, or landfill trash item
Reverse Vending Machine	A device that accepts empty used beverage containers and returns money to the user
Recyclable	Ability of an item to be recycled
UPC	Machine scannable barcode found on many consumer products or packaging
Magnetic Stripe Reader	Device that decodes information stored in the magnetic stripe of a card (e.g. credit card)
Coin Counter	Device that counts the value of the coins inserted
Open source	Production and development model that promote access to the source of the final product's source materials, such as the source code for a software product
End-user	A person who is not involved in the development/maintenance of an ELARA kiosk, who but operates the software. End-users are not expected to have technical expertise

Kiosk Manager/Administrator	Registered user who sets up and manages his/her own ELARA kiosk. Kiosk Managers are expected to have some technical expertise
GUI	Graphical User Interface
USB	Universal Serial Bus; industry standard for the connection of computer peripherals
JDK	Java Development Kit
JRE	Java Runtime Environment
OS	Operating System (e.g. Windows7, Linux, Android, iOS)
Http(s)	Hyper-text transfer protocol (secure)
SSH	Secure Shell network protocol
MVC	Model-View-Controller; Software engineering model

A.1.4 References

- MySQL web site: `www.mysql.org`
- PHP web site: `www.php.org`
- Wikipedia: `http://en.wikipedia.org/`
- SRS template: `http://www.cs.toronto.edu/~sme/CSC444F/assignments/CSC444-2001-SRS-01B.pdf`
- UML diagrams: `http://www.tracemolder.com/articles/a_quick_i ntroduction_to_uml_sequence_diagrams`

A.1.5 Overview of the Document

This document targets two different groups of readers: (1) software developers interested in extending and/or modifying the ELARA System, (2) Kiosk managers interested in becoming part of the ELARA project.

The rest of this document is organized as follows. In section A.2, the ELARA system is described as a whole in detail, and in section A.3, the features of the specific software requirements for the ELARA kiosk and the server are elaborated.

A.2 Overall Description

A.2.1 Product Perspective

The idea for an ELARA kiosk originated in Spring 2009 in an Embedded Computing course at Hobart and William Smith Colleges, Geneva, NY as a student final project. A very basic prototype of a kiosk was built on top of an Arduino Duemialnove platform, but its functionality was very limited.

During the Summer 2011, this same student, began working on an extension of this initial project. Out of this work originated the ELARA project with the ELARA kiosk being one of the major components of the hosting system.

The ELARA system is comprised of a central server which services requests from ELARA kiosks creating the ELARA kiosk - server subsystem, as well as hosting the ELARA website. The central server runs PHP scripts, but it also contains a MySQL database which stores all the information about recyclable items, registered users and each ELARA kiosk in the network. The ELARA kiosk then communicates with the server via Http/Https requests asking the server to perform a query to the database, do some computation, or to store information in the database. The ELARA website is the place for the crowd to contribute to the project and where each participant's recycling progress is recorded and available for display.

A.2.2 Product Functions

A.2.2.1 Kiosk

The design of an ELARA kiosk is based on the design of reverse vending machine since it also accepts recyclable and landfill waste from the user. However, an ELARA kiosk differs significantly from a reverse vending machine in that it does not give the user money back for the items they recycled, and it does not passively accept the waste item, but rather asks the user to dispose of it in the appropriate bin herself. This is the pedagogical aspect and the benefaction factor of the ELARA kiosk.

An ELARA kiosk has the following major features, which shall be explained in further detail in subsequent sections of this SRS:

- on site user authentication for recycling/contribution etc.
- item recognition via scan or manual entry of barcode
- item recognition through shape and material
- engage user in decision-making about recycling process
- indicate which bin the user should put her item in
- allow for session continuation or end of session
- display individuals recycling progress

Using these features at an ELARA kiosk is called a transaction. A session is multiple transactions in a single authentication period. This is explained in further detail in section A.3.2.1.

A.2.2.2 Server

The ELARA server has the following major features in relation to the kiosk-server subsystem, which shall be explained in further detail in subsequent sections of this SRS:

- authenticate kiosk
- authenticate the user and identify the recycling items via the received barcode
- identify if requested item is recyclable or landfill garbage and report this status
- evaluate user's choice of an unknown item's shape and material
- update the user's knowledge and recycling scores
- calculate user's total recycling progress

A.2.3 User Classes and Characteristics

A.2.3.1 Kiosk

There are three classes of users who use the ELARA kiosk. The first user class are the guests, people who only use the kiosk on rare occasions to ask for assistance in separating their recyclables from their landfill waste. These users do not have ongoing accounts on the ELARA website and their only contact with the ELARA system will be through casual use of the machine. The second class of users is the group of registered users. This group will form the ELARA community as they will have ongoing accounts on the ELARA website. Registered users share interactions with the kiosk with the guests, with the addition of personalized feedback on their recycling progress.

The third user class are the kiosk managers, namely the administrators who set up a kiosk, manage its use and perform scheduled hardware and software maintenance. These users should have technical expertise to the extent that they could modify the software, or exchange/repair parts of the hardware if need be. This class of users is a subset of the registered users class since only registered users may manage a kiosk.

A.2.3.2 Server

There is only one class of users who will have access to the ELARA server, namely the ELARA system administrators. They will have high technical expertise in working with databases, networking, security as well as server scripting.

A.2.4 Operating Environment

An ELARA kiosk is a nearly-autonomous machine. The only instance in which the machine is controlled externally is when its kiosk manager is performing hardware maintenance or repairs on the kiosk, or when its kiosk manager logs into the machine

remotely via ssh. Otherwise, nobody has access to the back of the kiosk. The recycling and garbage bins belonging to the kiosk are ideally located within a lockable container as part of the kiosk. Cleaning staff of the institution the kiosk is part of and the kiosk manager are the only persons to have access to the bins to empty or exchange them.

Since an ELARA kiosk is predominantly self-sufficient it can remain unsupervised during operation hours. However, since the building materials of the kiosk are not necessarily weather proof, it is not suggested that the machine be placed outside.

Furthermore, the kiosk requires a 110V electrical socket connection to operate and access to a wireless internet connection.

A.2.5 Design and Implementation Constraints

There are many constraints to consider for the ELARA kiosk server subsystem described in this document. First and foremost, this system was designed, implemented and tested under a significant human resource constraint, namely by a single undergraduate computer science student during a single academic year. This also means that the system was created under the time constraint of having to deliver the system in under less than nine months.

Moreover, since the ELARA project is meant to be an open source project, the cost of the equipment used to build a single kiosk had to be kept under $1000 per kiosk to render joining the ELARA system more accessible and attractive for any person who wishes to join and is working with a low budget. With this in mind, the kiosk was also designed to be easy to maintain by its kiosk manager. subsectionAssumptions and Dependencies One major assumption is that most recyclable items have a UPC barcode on them in some form (more commonly a 1-D barcode). In order to deal with items which do not have a barcode the kiosk will have a short and easy interactive process to assess the shape and material of the item to make a more accurate assessment of whether or not the item is recyclable.

It is also assumed that any software developer working to create her own version of the ELARA kiosk software will have different time and resource constraints. For this reason, a clear distinction between required functions and optional features will be made in this document. The assumption is also made that the original developer of the ELARA kiosk prototype is pressed for time such that only the absolute minimum requirements the kiosk must have are implemented. Functions labeled as additional are optional and their development is at the discretion of the machine managers. Further assumptions include that users be sighted and able to read (and potentially touch) the display, and that the ELARA kiosk computer is secure and protected to user tampering.

A.3 Specific Requirements

A.3.1 External Interface Requirements

A.3.1.1 User Interfaces

There are two types of user interfaces for an ELARA kiosk. The user may interact via a GUI with a touch screen computer and a smart phone/tablet PC, while the interface for the personal computer is text. The user interacting with the GUI will be able to press virtual buttons appearing on the screen to make selections, enter information into the kiosk, etc. On the other hand, a user using the text-based program can enter information via the keyboard and the mouse/touch pad. Since the server is only accessible to the system administrator, the user interface to the server depends on how the system administrator communicates with it.

A.3.1.2 Hardware Interfaces

One requirement for the ELARA kiosk hardware is that it has either sufficient USB ports for all the required peripherals, or that it contains at least two USB ports which can host a USB hub for multiple USB devices. This implies that any peripheral that is attached to the kiosk must be USB compatible. Furthermore, the kiosk must allow for an internet connection, preferably through a wireless internet connection for mobility purposes.

Another requirement is that the kiosk have a monitor or screen that users can read and potentially touch as this is how they interact with the machine. At the same time, if the kiosk does not have a touch screen panel, it is required to include a keyboard and a mouse or touch pad as the second means of entering information into the ELARA kiosk.

A.3.1.3 Software Interfaces

Kiosk The kiosk makes use of operating system calls to the device file management system to retrieve its machine ID and its cryptographically hashed digital signature from files on disk.

An ELARA kiosk communicates only with the server via the HTTP protocol. Incoming data into the kiosk are therefore server responses containing a response code which depends on the current state of the kiosk and of the information requested.

Server The requirements for the software interfaces on the server side of the subsystem are that it supports MySQL, and PHP. An Apache server of the latest version of Apache is recommended and preferred.

A.3.1.4 Communications Interfaces

An ELARA kiosk shall be connected to the internet via an Ethernet port on the kiosk machine, or a wireless router. Furthermore, the kiosk shall send messages of a specific

format to the server, which the server scripts will process appropriately.

All the messages sent between the kiosk and the server must follow the Http or Https protocol, and in particular must be formatted such that they can be properly interpreted by the PHP POST method. The message digest sent along with the message is used to properly authenticate each kiosk on the server-side, and at the same time, authenticate the server on the client-side. This prevents many security breaches at the network level. An alternative to this type of authentication would be a similar concept as the Google Maps API key.

A.3.2 Functional Requirements of an ELARA kiosk

A.3.2.1 User authentication

On site End-User authentication for recycling/contribution Since there are three types of end-users, user authentication will be slightly different for each enduser class. The fundamental requirement is that the distinction between guest users and registered users is made; i.e., that the user standing in front of the kiosk is able to indicate whether or not they will scan validate their identity. Being able to distinguish between guest users and registered users allows for the data of a guest to be recorded in the database for documenting the recycling progress of the general population. Registered users set themselves apart from guests because of the former class of users must additionally enter their username besides indicating their user class at the beginning of their session.

If the user indicates she is a registered user, but fails to authenticate herself properly (e.g. wrong username, or nonexistent registered user), she will have to restart the authentication process in order to use the machine.

A registered user cannot proceed to perform a transaction on the machine unless she indicates her user class and authenticates successfully. However, for any user class, once the user is authenticated, she may perform multiple transactions without having to log in every time she begins a new transaction.

The information about user class and username is sent to the ELARA server via an Http/Https request.

Machine Manager authentication Machine Managers are not be able to authenticate to work on the machine using the user interface of the kiosk application. Instead, they must log into the machine remotely and then maintain the software and hardware. Since the kiosk is networked, remote login with ssh, or any other secure protocol is the required method of accessing the machine remotely.

A.3.2.2 Item recognition via scan or manual entry of barcode

Since it is assumed that many items end-users will try to use the kiosk for have a barcode on them, the ELARA kiosk must contain a means of entering/reading in an item's barcode. There are three methods for entering in a barcode: (1) by means of a UPC barcode scanner, which would to allow the users to easily identify their item

for the machine in the second step of a transaction, (2) using some sort of keyboard or keypad so the user may enter her item's barcode manually if a UPC scanner is not available, or (3) via a barcode recognition application program which can decode a picture of a barcode into the actual value of the UPC. While the UPC scanner method is preferred for ease-of-use purposes, the main requirement remains that the kiosk provide a means of identifying the item.

After the user has been authenticated successfully, this barcode is sent to the server.

A.3.2.3 Item recognition through its shape and material

Should an item not have a barcode, not have an identifying record in the database, or its recyclability be unclear, the user must be given a quick and easy process to identify the item by indicating its general shape (such as bottle, cylindrical, spherical etc.), and its composition material such that the machine can make a more valid assessment of the recyclability of the item. The user's choices are sent to the server for evaluation.

A.3.2.4 Engage user in decision-making about the recycling process

The purpose of this requirement is merely an educational and statistical one to educate the user about the choices she makes when she separates her garbage, while also recording her knowledge about recycling. This requirement complies with the scope of the ELARA project stated in section A.1.2.

If the server has successfully identified a scanned item as recyclable or not (see section A.3.3.3), the kiosk will prompt the user to answer whether or not she believes her item is recyclable. The kiosk will send the user's answer to the server to record it in the database. Otherwise the process described in section A.3.2.3 will take place.

A.3.2.5 Correct bin indication

After asking the user to identify her item, or recognizing an item through its shape and material, the kiosk must indicate whether or not they answered correctly and more importantly which bin their item belongs. The user will be asked to insert her item in the appropriate bin.

A.3.2.6 Session Continuation

Users must be allowed to continue with another transaction, which will bring them back to the item recognition step of a transaction, or end their session.

A.3.2.7 Display recycling progress

At the end of a user's session, regardless if registered or not, the user's recycling progress is to be displayed at the end of each transaction as an incentive and motivation her. For guests the recycling progress may reflect the recycling progress in

the past month for all guests at that particular kiosk, institution or town, and for registered users this figure displayed will be personalized. What this means is that the registered user's recycling score (a count of how many items they have recycled in a given time frame) is to be shown to her at the kiosk for her to assess her progress in relation to how much has been recycled in total at the particular kiosk.

A.3.3 Functional Requirements of the ELARA server

A.3.3.1 Kiosk authentication

Upon the reception of every Http/Https request sent from the kiosk, the server will authenticate the kiosk to prevent man-in-the-middle attacks. This can be done by way of a cryptographic hash of the entire message sent to the server, which includes the kiosk's unique digital signature. The server must then retrieve the kiosk's digital signature it has stored in the database, create its own cryptographic hash of the entire message with its version of the digital signature, and then compare the hashes. Should the hashes not match up, the server will respond indicating a failed kiosk authentication. The other component of kiosk authentication is the timestamp included in each message sent from the kiosk as well. This timestamp will help prevent replay attacks.

A.3.3.2 User authentication

When the kiosk is in the first stage of a transaction, as is described in section A.3.2.1, the server will receive a request from the kiosk containing the user class and potentially the username of the current user if she has indicated she is a registered user. If a username was included in the message, the server must authenticate this user by way of querying the database for the username. If it cannot be found, the server reports failure. Otherwise, the server reports back the user's unique user ID (0 for guests, and greater than 0 for registered users) with which the ELARA kiosk server subsystem can open a session with that particular user.

A.3.3.3 Identify requested item and determine if recyclable

During the second stage of a transaction (see section A.3.2.2), the server will receive a request containing an item barcode. The server then queries the database for this barcode and the respective item's recyclability. If the item is identified as recyclable, or as landfill garbage, the server responds with this status. The step specified in section A.3.2.4 is then initiated. Otherwise, if the item is unknown or if it's recyclability cannot be determined with the information stored about it in the database, the server response indicates failure to identify the item. In this case, the process described in section A.3.2.3 takes place.

A.3.3.4 Evaluate user's choice of unknown item's shape and material

In the case that the user's item does not have a barcode, or the server fails to identify the item, the server will receive a message from the kiosk indicating which shape and composition material the user has chosen for her item. Based upon these two pieces of information, the ELARA server will make a more knowledgeable assessment about whether or not the user's item is recyclable. For instance, if the user indicates, the item is cylindrical and made of glass, the server will recognize glass as a recyclable material and thus respond to the kiosk with its estimation.

A.3.3.5 Update registered user's knowledge and recycling scores

If the user is registered with the ELARA system, the answer to her decision about her item (see section A.3.2.4) will be sent to the server for her recycling score to be updated. This means, if she guessed correctly that her item is recyclable her knowledge score will increase to indicate how knowledgeable about recycling she is. In addition, regardless of her answer, if the item is recyclable, her recycling score must go up.

A.3.3.6 Calculate user's total recycling progress

Once a user has indicated that she is ending her session at the kiosk, the server will respond to this message with information about the user's recycling progress. If the user is a guest, then the total number of items recycled by all guests at that particular kiosk is returned. Otherwise, the registered user's personalized recycling progress, namely a figure reflecting how much this user has recycled within a given time frame, is returned to the kiosk.

A.3.4 Design Constraints

All application software written for any platform for the ELARA kiosk must be written in a programming language which supports networking capabilities. Examples of such languages include, Java, C++, C, and any other language which can easily support all the requirements. In the prototype version of the ELARA kiosk (implemented on an LCD Touch Panel Computer), the Java programming language is chosen.

A.3.5 Software System Attributes

A.3.5.1 Reliability

The ELARA kiosk software shall never crash or hang, unless an operating system error occurs. The software shall provide the necessary measures, such as resetting the session if a the network times out, to deal with network delays and failures.

A.3.5.2 Availability

An ELARA kiosk shall be available to any member of the hosting institution of the system provided the institution complies with the user authentication requirements stated in section A.3.2.1 of this document.

The kiosk are available 24 hours per day, unless when they are undergoing administrative maintenance.

A.3.5.3 Security and Privacy

The ELARA kiosk will provide appropriate security measures, namely encryption or cryptographic hashing of the information entered, for user authentication depending on the chosen method of authentication (ID card with barcode, ID card with magnetic stripe, RFID tag). No confidential information will be required of the users, and no other information contained within the authentication devices shall be disclosed to any party other than the ELARA project.

A.3.5.4 Maintainability

All provided source code shall be fully documented. Each function contained within the source code shall include pre- and post-conditions. All source code files shall contain information about authorship and about updates made to the code.

All source code shall be modular to allow for future modifications, and it is maintained via a version control system such as CVS or Subversion.

A.3.5.5 Portability

The software shall be supported on at least one of three following types of platforms:

- LCD Touch Panel Computer as described above (main platform)
- Personal Computer (desk- or laptop) supporting the JDK
- Smart Phones running any version of Android OS

A.3.5.6 Safety

There are no specific safety requirements.

A.3.5.7 Training-related Requirements

No specific training should be necessary for an end-user to operate an ELARA kiosk. Any kiosk manager should have some basic knowledge in computer hardware, and proficient knowledge of the Java programming language.

A.3.5.8 Packaging Requirements

The source code for the ELARA kiosk (touch panel, and personal computer versions) will be available for download as a single compressed file from Source Forge. The uncompressed set of files include a README file describing the process for installing and running the software.

A.3.5.9 Legal Requirements

Copyright laws and license agreements must be respected for any piece of software of the given source code being used and/or modified, as well as for any third party software used to enhance the ELARA kiosk software. All ELARA software source code is copyright by Marcela Melara.

A.3.6 Optional Features

The following features of the kiosk software are optional:

- More interactive and entertaining user redemption by giving the option to take a picture of the user if she user has just recycled the 100th item of one kind in a particular time frame (guest users), or reached some other milestone in their recycling progress (registered users). The picture can then be displayed on the ELARA website or published in an institution-specific publication.
- Users receive some sort of monetary redemption in the form of a printed coupon
- Users have the choice to donate the value of their recycled items to a charitable cause specified by the kiosk (dependent on institution collaborations and affiliations)
- Users may donate spare coin change to a charitable cause specified by the kiosk (dependent on institution collaborations and affiliations)

A.3.7 Other Requirements

There are no other requirements.

Bibliography

[1] R. Ramakrishnan A. Doan and A. Y. Halevy. Crowdsourcing systems on the world-wide web. *Communications of the ACM*, 54(4):86–96, Apr 2011.

[2] Arduino. Arduino, n.d. [Online; accessed 8-April-2012].

[3] ASP.net4.com. Model view controller, Jun 2011. [Online; accessed 24-March-2012].

[4] U.S. EPA. Municipal solid waste generation, recycling and disposal in the united states: Facts and figures for 2010.

[5] S. Greengard. Following the crowd. *Communications of the ACM*, 54(2):20–22, Feb 2011.

[6] J. Howe. *Crowdsourcing: Why the Power of the Crowd is Driving the Future of Business*. Crown Publishing Group, New York, NY, 2008.

[7] U. Bretschneider J. M. Leitmeister, M. Huber and H. Krcmar. Leveraging crowdsourcing: Activation-supporting components for it-based ideas competitions. *Journal of Management Informations Systems*, 26(1):197–224, 2009.

[8] Envirobank Recycling Pty Ltd. Video: How to recycle for rewards - envirobank 7-eleven launch july 2011, 2011. [Online; accessed 23-March-2012].

[9] L. L. Peterson and B. S. Davie. *Computer Networks: A Systems Approach, 4th. edn.* Morgan Kaufmann, San Franciscon, CA, 2007.

[10] Hal Philipp. Please touch. explore the evolving world of touchscreen technology, April 2008. [Online; accessed 8-April-2012].

[11] Tomra Systems SAS. Our organization - the beginning, n.d. [Online; accessed 20-March-2012].

[12] Technologic Systems. Ts-tcp-8900, 2011. [Online; accessed 8-April-2012].

[13] TutorialsPoint.com. Client server model - architecture, n.d. [Online; accessed 28-March-2012].

[14] W3Schools.com. The xmlhttprequest object, n.d. [Online; accessed 26-March-2012].

[15] Wikipedia. Cryptographic hash function — wikipedia , the free encylcopedia, 2012. [Online; accessed 28-March-2012].

[16] Wikipedia. Finite-state machine — wikipedia , the free encylcopedia, 2012. [Online; accessed 30-March-2012].

[17] Wikipedia. Man-in-the-middle attack — wikipedia , the free encylcopedia, 2012. [Online; accessed 28-March-2012].

[18] Wikipedia. Microcontroller — wikipedia , the free encylcopedia, 2012. [Online; accessed 8-April-2012].

[19] Wikipedia. Reverse vending machine — wikipedia , the free encylcopedia, 2012. [Online; accessed 22-March-2012].